ACM Template

UESTC_Lasagne

Last build at October 16, 2012

Contents

1 To Do List

所有带*的内容。。。

可以从原来的模板里面继承一些好东西过来。

set,map,multiset等的搞基用法，以及注意事项。

生成树计数

2 注意事项

10^6数量级慎用后缀数组

TLE的时候要冷静哟。。

思考的时候结合具体步骤来的话 会体会到一些不同的东西

C++与G++是很不一样的。。。

map套字符串是很慢的。。。

栈会被记录内存。。。

浮点数最短路要注意取≤来判断更新。。。

注意 long long

不要相信.size()

重复利用数组时 小心数组范围

先构思代码框架 每当实际拍马框架变化时 停手 重新思考

有时候四边形不等式也是帮得上忙的 dp 优化是可以水的

结构体里面带数组会非常慢,有时候 BFS 把数组压成数字会快很多。

```
1 void fun(int a[])
2 {
3     printf("%d\n",sizeof(a));
4 }
```

结果是 sizeof(a[0]),如果传数组指针然后要清空的话不要用 sizeof。

sqrt 某些时候会出现 sqrt(-0.00)的问题。

将code::blocks的默认终端改成gnome-terminal

```
1 gnome-terminal -t $TITLE -x
```

最小割割集找法 在残量网络中从源点出发能到的点集记为S原图中S到S'的边即是最小割集

double全局变量初始值可能不是0

3 字符串处理

3.1 *AC自动机

3.1.1 指针

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex
{
    Vertex *fail,*next[CHAR];
    Vertex(){}
    Vertex(bool flag)//为什么要这样写?
    {
        fail=0;
        memset(next,0,sizeof(next));
    }
};
int size;
Vertex vertex[TOTLEN+1];
void init()
{
    vertex[0]=Vertex(0);
    size=1;
}
void add(Vertex *pos,int cha)
{
    vertex[size]=Vertex(0);
    pos->next[cha]=&vertex[size++];
}
void add(vector<int> s)
{
    int l=s.size();
    Vertex *pos=&vertex[0];
    for (int i=0; i<l; i++)
    {
        if (pos->next[s[i]]==NULL)
            add(pos,s[i]);
        pos=pos->next[s[i]];
    }
}
void bfs()
{
    queue<Vertex *> que;
    Vertex *u=&vertex[0];
    for (int i=0; i<CHAR; i++)
        if (u->next[i]!=NULL)
        {
```

```
44              que.push(u->next[i]);
45              u->next[i]->fail=u;
46          }
47          else
48              u->next[i]=u;
49      u->fail=NULL;
50      while (!que.empty())
51      {
52          u=que.front();
53          que.pop();
54          for (int i=0; i<CHAR; i++)
55              if (u->next[i]!=NULL)
56              {
57                  que.push(u->next[i]);
58                  u->next[i]->fail=u->fail->next[i];
59              }
60              else
61                  u->next[i]=u->fail->next[i];
62      }
63  }
```

3.1.2 非指针

```
1   struct Trie
2   {
3       int next[50][10],fail[50];
4       bool end[50];
5       int L,root;
6
7       int newNode()
8       {
9           for (int i = 0;i < 10;i++)
10              next[L][i] = -1;
11          end[L] = false;
12          return L++;
13      }
14
15      void Init()
16      {
17          L = 0;
18          root = newNode();
19      }
20
21      void Insert(char s[])
22      {
23          int now = root;
24          for (int i = 0;s[i] != 0;i++)
25          {
26              if (next[now][s[i]-'0'] == -1)
```

```
                next[now][s[i]-'0'] = newNode();
            now = next[now][s[i]-'0'];
        }
        end[now] = true;
    }

    void Build()
    {
        queue<int> Q;
        for (int i = 0;i < 10;i++)
            if (next[root][i] == -1)
                next[root][i] = root;
            else
            {
                fail[next[root][i]] = root;
                Q.push(next[root][i]);
            }
        while (!Q.empty())
        {
            int now = Q.front();
            Q.pop();
            end[now] |= end[fail[now]];
            for (int i = 0;i < 10;i++)
                if (next[now][i] == -1)
                    next[now][i] = next[fail[now]][i];
                else
                {
                    fail[next[now][i]] = next[fail[now]][i];
                    Q.push(next[now][i]);
                }
        }
    }
};
```

3.2 后缀数组

3.2.1 DC3

所有下标都是0 n-1，height[0]无意义。

```
//所有相关数组都要开三倍
const int maxn = 300010;
# define F(x) ((x)/3+((x)%3==1?0:tb))
# define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
int c0(int *r, int a, int b)
{
    return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[
       b + 2];
}
```

```
10 int c12(int k, int *r, int a, int b)
11 {
12     if (k == 2) return r[a] < r[b] || r[a] == r[b] && c12(1, r, a
           + 1, b + 1);
13     else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b +
           1];
14 }
15 void sort(int *r, int *a, int *b, int n, int m)
16 {
17     int i;
18     for (i = 0; i < n; i++) wv[i] = r[a[i]];
19     for (i = 0; i < m; i++) ws[i] = 0;
20     for (i = 0; i < n; i++) ws[wv[i]]++;
21     for (i = 1; i < m; i++) ws[i] += ws[i - 1];
22     for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
23     return;
24 }
25 void dc3(int *r, int *sa, int n, int m)
26 {
27     int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) /
          3, tbc = 0, p;
28     r[n] = r[n + 1] = 0;
29     for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
30     sort(r + 2, wa, wb, tbc, m);
31     sort(r + 1, wb, wa, tbc, m);
32     sort(r, wa, wb, tbc, m);
33     for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
34         rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
35     if (p < tbc) dc3(rn, san, tbc, p);
36     else for (i = 0; i < tbc; i++) san[rn[i]] = i;
37     for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i]
          * 3;
38     if (n % 3 == 1) wb[ta++] = n - 1;
39     sort(r, wb, wa, ta, m);
40     for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
41     for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
42         sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j
              ++];
43     for (; i < ta; p++) sa[p] = wa[i++];
44     for (; j < tbc; p++) sa[p] = wb[j++];
45 }
46 //str和sa也要三倍
47 void da(int str[], int sa[], int rank[], int height[], int n, int
       m)
48 {
49     for (int i = n; i < n * 3; i++)
50         str[i] = 0;
51     dc3 (str , sa , n + 1 , m);
```

```
52      int i, j, k;
53      for (i = 0; i < n; i++)
54      {
55          sa[i] = sa[i + 1];
56          rank[sa[i]] = i;
57      }
58      for (i = 0, j = 0, k = 0; i < n; height[rank[i ++]] = k)
59          if (rank[i] > 0)
60              for (k ? k-- : 0 , j = sa[rank[i] - 1]; i + k < n &&
                    j + k < n &&
61                        str[i + k] == str[j + k]; k ++) ;
62  }
```

3.2.2 DA

这份似乎就没啥要注意的了。

```
1   const int maxn = 200010;
2   int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
3
4   bool cmp(int *r,int n,int a,int b,int l)
5   {
6       return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
7   }
8   void da(int str[],int sa[],int rank[],int height[],int n,int m)
9   {
10      int *s = str;
11      int *x=wx,*y=wy,*t,p;
12      int i,j;
13      for(i=0; i<m; i++)wss[i]=0;
14      for(i=0; i<n; i++)wss[x[i]=s[i]]++;
15      for(i=1; i<m; i++)wss[i]+=wss[i-1];
16      for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
17      for(j=1,p=1; p<n && j<n; j*=2,m=p)
18      {
19          for(i=n-j,p=0; i<n; i++)y[p++]=i;
20          for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
21          for(i=0; i<n; i++)wv[i]=x[y[i]];
22          for(i=0; i<m; i++)wss[i]=0;
23          for(i=0; i<n; i++)wss[wv[i]]++;
24          for(i=1; i<m; i++)wss[i]+=wss[i-1];
25          for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
26          for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
27              x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
28      }
29      for(int i=0; i<n; i++) rank[sa[i]]=i;
30      for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
31          if(rank[i]>0)
```

```
32              for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n &&
                   str[i+k]==str[j+k]; k++);
33  }
```

3.3 后缀三兄弟

```
1   #include <cstdio>
2   #include <cstring>
3   #include <algorithm>
4   using namespace std;
5   const int CHAR = 26;
6   const int MAXN = 100000;
7   struct SAM_Node
8   {
9       SAM_Node *fa,*next[CHAR];
10      int len;
11      int id,pos;
12      SAM_Node() {}
13      SAM_Node(int _len)
14      {
15          fa = 0;
16          len = _len;
17          memset(next,0,sizeof(next));
18      }
19  };
20  SAM_Node SAM_node[MAXN * 2], *SAM_root, *SAM_last;
21  int SAM_size;
22  SAM_Node *newSAM_Node(int len)
23  {
24      SAM_node[SAM_size] = SAM_Node(len);
25      SAM_node[SAM_size].id=SAM_size;
26      return &SAM_node[SAM_size++];
27  }
28  SAM_Node *newSAM_Node(SAM_Node *p)
29  {
30      SAM_node[SAM_size] = *p;
31      SAM_node[SAM_size].id=SAM_size;
32      return &SAM_node[SAM_size++];
33  }
34  void SAM_init()
35  {
36      SAM_size = 0;
37      SAM_root = SAM_last = newSAM_Node(0);
38      SAM_node[0].pos=0;
39  }
40  void SAM_add(int x,int len)
41  {
42      SAM_Node *p = SAM_last, *np = newSAM_Node(p->len + 1);
```

```
43      np->pos=len;
44      SAM_last = np;
45      for (; p && !p->next[x]; p = p->fa)
46          p->next[x] = np;
47      if (!p)
48      {
49          np->fa = SAM_root;
50          return ;
51      }
52      SAM_Node *q = p->next[x];
53      if (q->len == p->len + 1)
54      {
55          np->fa = q;
56          return ;
57      }
58      SAM_Node *nq = newSAM_Node(q);
59      nq->len = p->len + 1;
60      q->fa = nq;
61      np->fa = nq;
62      for (; p && p->next[x] == q; p = p->fa)
63          p->next[x] = nq;
64  }
65  void SAM_build(char *s)
66  {
67      SAM_init();
68      int l = strlen(s);
69      for (int i = 0; i < l; i++)
70          SAM_add(s[i] - 'a',i+1);
71  }
72
73  SAM_Node * SAM_add(SAM_Node *p, int x, int len)
74  {
75      SAM_Node *np = newSAM_Node(p->len + 1);
76      np->pos = len;
77      SAM_last = np;
78      for (; p && !p->next[x]; p = p->fa)
79          p->next[x] = np;
80      if (!p)
81      {
82          np->fa = SAM_root;
83          return np;
84      }
85      SAM_Node *q = p->next[x];
86      if (q->len == p->len + 1)
87      {
88          np->fa = q;
89          return np;
90      }
```

```
91      SAM_Node *nq = newSAM_Node(q);
92      nq->len = p->len + 1;
93      q->fa = nq;
94      np->fa = nq;
95      for (; p && p->next[x] == q; p = p->fa)
96          p->next[x] = nq;
97      return np;
98  }
99  void SAM_build(char *s)//多串建立 注意SAM_init()的调用
100 {
101     int l = strlen(s);
102     SAM_Node *p = SAM_root;
103     for (int i = 0; i < l; i++)
104     {
105         if (!p->next[s[i] - 'a'] || !(p->next[s[i] - 'a']->len ==
                i + 1))
106             p=SAM_add(p,s[i] - 'a', i + 1);
107         else
108             p = p->next[s[i] - 'a'];
109     }
110 }
111
112 struct ST_Node
113 {
114     ST_Node *next[CHAR],*fa;
115     int len,pos;
116 }ST_node[MAXN*2],*ST_root;
117 int Sufpos[MAXN];
118 void ST_add(int u,int v,int chr,int len)
119 {
120     ST_node[u].next[chr]=&ST_node[v];
121     ST_node[v].len=len;
122 }
123 void init(int n)
124 {
125     for (int i=0;i<n;i++)
126     {
127         ST_node[i].pos=-1;
128         ST_node[i].fa=0;
129         memset(ST_node[i].next,0,sizeof(ST_node[i].next));
130     }
131     ST_node[0].pos=0;
132     ST_root=&ST_node[0];
133 }
134 void ST_build(char *s)
135 {
136     int n=strlen(s);
137     reverse(s,s+n);
```

```
138     SAM_build(s);
139     init(SAM_size);
140     for (int i=1;i<SAM_size;i++)
141     {
142         ST_add(SAM_node[i].fa->id,SAM_node[i].id,s[SAM_node[i].
               pos-SAM_node[i].fa->len-1]-'a',SAM_node[i].len-
               SAM_node[i].fa->len);
143         if (SAM_node[i].pos==SAM_node[i].len)
144         {
145             Sufpos[n-SAM_node[i].pos+1]=i;
146             ST_node[i].pos=n-SAM_node[i].pos+1;
147         }
148     }
149 }
150
151 int rank[MAXN],sa[MAXN+1];
152 int height[MAXN];
153 int L;
154 void ST_dfs(ST_Node *p)
155 {
156     if (p->pos!=-1)
157         sa[L++]=p->pos;
158     for (int i=0;i<CHAR;i++)
159         if (p->next[i])
160             ST_dfs(p->next[i]);
161 }
162 char s[MAXN+1];
163 int main()
164 {
165     gets(s);
166     ST_build(s);
167     L=0;
168     ST_dfs(ST_root);
169     int n=strlen(s);
170     for (int i=0; i<n; i++)
171         sa[i]=sa[i+1]-1;
172     for (int i=0; i<n; i++)
173         rank[sa[i]]=i;
174     reverse(s,s+n);
175     for (int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
176         if (rank[i])
177             for (k?k--:0,j=sa[rank[i]-1]; s[i+k]==s[j+k]; k++);
178 }
```

3.3.1 例题

```
1 #include <iostream>
2 #include <algorithm>
3 #include <cstdio>
```

```
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {
19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
42     SAM_root = SAM_last = newSAM_Node(0);
43 }
44 void SAM_add(int x,int len)
45 {
46     SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47     SAM_last = np;
48     for (; p&&!p->next[x]; p=p->fa)
49         p->next[x] = np;
50     if (!p)
51     {
```

```
52          np->fa = SAM_root;
53          return;
54      }
55      SAM_Node *q = p->next[x];
56      if (q->len == p->len+1)
57      {
58          np->fa = q;
59          return;
60      }
61      SAM_Node *nq = newSAM_Node(q);
62      nq->len = p->len+1;
63      q->fa = nq;
64      np->fa = nq;
65      for (; p&&p->next[x] == q; p = p->fa)
66          p->next[x] = nq;
67  }
68  int getid(char ch)
69  {
70      return ch-'a';
71  }
72  void SAM_build(char *s)
73  {
74      SAM_init();
75      int l = strlen(s);
76      for (int i = 0; i < l; i++)
77          SAM_add(getid(s[i]),i+1);
78  }
79  char s[10][MAXN+1];
80  int ans;
81  int head[MAXN*2];
82  struct Edge
83  {
84      int to,next;
85  } edge[MAXN*2];
86  int M;
87  int n;
88  void add_edge(int u,int v)
89  {
90      edge[M].to=v;
91      edge[M].next=head[u];
92      head[u]=M++;
93  }
94  void dfs(int u)
95  {
96      for (int i=head[u]; i!=-1; i=edge[i].next)
97      {
98          int v=edge[i].to;
99          dfs(v);
```

```
100         for (int j=0; j<n-1; j++)
101             SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
                    ].mat[j]);
102     }
103     int tmp=SAM_node[u].len;
104     for (int i=0; i<n-1; i++)
105         tmp=min(tmp,SAM_node[u].mat[i]);
106     ans=max(ans,tmp);
107 }
108 int main()
109 {
110
111     while (scanf("%s",s[n])!=EOF)
112         n++;
113     int L=strlen(s[0]);
114     ans=M=0;
115     SAM_build(s[0]);
116     for (int j=1; j<n; j++)
117     {
118         int l=strlen(s[j]),len=0;
119         SAM_Node *p=SAM_root;
120         for (int i=0; i<l; i++)
121         {
122             if (p->next[getid(s[j][i])])
123             {
124                 p=p->next[getid(s[j][i])];
125                 p->mat[j-1]=max(p->mat[j-1],++len);
126             }
127             else
128             {
129                 while (p && !p->next[getid(s[j][i])])
130                     p=p->fa;
131                 if (!p)
132                 {
133                     p=SAM_root;
134                     len=0;
135                 }
136                 else
137                 {
138                     len=p->len+1;
139                     p=p->next[getid(s[j][i])];
140                 }
141                 p->mat[j-1]=max(p->mat[j-1],len);
142             }
143         }
144     }
145     memset(head,-1,4*SAM_size);
146     for (int i=1; i<SAM_size; i++)
```

```
147         add_edge(SAM_node[i].fa->id,i);
148     dfs(0);
149     printf("%d\n",ans);
150     return 0;
151 }
```

LCS2

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {
19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
```

```
42      SAM_root = SAM_last = newSAM_Node(0);
43  }
44  void SAM_add(int x,int len)
45  {
46      SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47      SAM_last = np;
48      for (; p&&!p->next[x]; p=p->fa)
49          p->next[x] = np;
50      if (!p)
51      {
52          np->fa = SAM_root;
53          return;
54      }
55      SAM_Node *q = p->next[x];
56      if (q->len == p->len+1)
57      {
58          np->fa = q;
59          return;
60      }
61      SAM_Node *nq = newSAM_Node(q);
62      nq->len = p->len+1;
63      q->fa = nq;
64      np->fa = nq;
65      for (; p&&p->next[x] == q; p = p->fa)
66          p->next[x] = nq;
67  }
68  int getid(char ch)
69  {
70      return ch-'a';
71  }
72  void SAM_build(char *s)
73  {
74      SAM_init();
75      int l = strlen(s);
76      for (int i = 0; i < l; i++)
77          SAM_add(getid(s[i]),i+1);
78  }
79  char s[MAXN+1];
80  int ans;
81  int head[MAXN*2];
82  struct Edge
83  {
84      int to,next;
85  } edge[MAXN*2];
86  int M;
87  int n;
88  void add_edge(int u,int v)
89  {
```

```
90      edge[M].to=v;
91      edge[M].next=head[u];
92      head[u]=M++;
93  }
94  void dfs(int u)
95  {
96      for (int i=head[u]; i!=-1; i=edge[i].next)
97      {
98          int v=edge[i].to;
99          /*for (int j=0; j<n; j++)
100             SAM_node[v].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
                ].mat[j]);*/
101         dfs(v);
102         for (int j=0; j<n; j++)
103             SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
                ].mat[j]);
104     }
105     int tmp=SAM_node[u].len;
106     for (int i=0; i<n; i++)
107         tmp=min(tmp,SAM_node[u].mat[i]);
108     ans=max(ans,tmp);
109 }
110 int main()
111 {
112     //freopen("in.txt","r",stdin);
113     //freopen("out.txt","w",stdout);
114     n=0;
115     gets(s);
116     SAM_build(s);
117     while (gets(s))
118     {
119         int l=strlen(s),len=0;
120         SAM_Node *p=SAM_root;
121         for (int i=0; i<l; i++)
122         {
123             if (p->next[getid(s[i])])
124             {
125                 p=p->next[getid(s[i])];
126                 p->mat[n]=max(p->mat[n],++len);
127             }
128             else
129             {
130                 while (p && !p->next[getid(s[i])])
131                     p=p->fa;
132                 if (!p)
133                 {
134                     p=SAM_root;
135                     len=0;
```

```
                }
                else
                {
                    len=p->len+1;
                    p=p->next[getid(s[i])];
                }
                p->mat[n]=max(p->mat[n],len);
            }
            //printf("%d %d %d\n",i,len,p->id);
        }
        n++;
    }
    memset(head,-1,4*SAM_size);
    for (int i=1; i<SAM_size; i++)
        add_edge(SAM_node[i].fa->id,i);
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

3.4 KMP

求A[0..i]的一个后缀最多能匹配B的前缀多长。 先对B进行自匹配然后与A匹配。 KMP[i]就是对应答案，p[i]+1是B[0..i]的一个后缀最多能匹配B的前缀多长。

```
//自匹配过程
int j;
p [0] = j = -1;
for ( int i = 1; i < lb; i++)
{
    while (j >= 0 && b[j + 1] != b[i]) j = p[j];
    if (b[j + 1] == b[i]) j ++;
    p[i] = j;
}
//下面是匹配过程
j = -1;
for ( int i = 0; i < la; i++)
{
    while (j >= 0 && b[j + 1] != a[i]) j = p[j];
    if (b[j + 1] == a[i]) j ++;
    KMP[i] = j + 1;
}
```

3.5 e-KMP

求A[i..len-1]和B的最长公共前缀有多长。 先对B进行自匹配然后与A匹配。 eKMP[i]就是对应答案。p[i]是B[i..len-1]和B的最长公共前缀有多长。

```
//自匹配过程
int j = 0;
```

```
while (j < lb && b[j] == b[j + 1])
    j++;
p[0] = lb, p[1] = j;
int k = 1;
for (int i = 2; i < lb; i++)
{
    int Len = k + p[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        p[i] = L;
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < lb && b[i + j] == b[j])
            j++;
        p[i] = j, k = i;
    }
}
//下面是匹配过程
j = 0;
while (j < la && j < lb && a[j] == b[j])
    j++;
eKMP[0] = j;
k = 0;
for (int i = 1; i < la; i++)
{
    int Len = k + eKMP[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        eKMP[i] = L;
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < la && j < lb && a[i + j] == b[j])
            j++;
        eKMP[i] = j, k = i;
    }
}
```

3.6 *Manacher

待整理

```
char s[1000],a[3000];
int p[3000],len,l,pnow,pid,res,resid;

int main()
{
    while (scanf("%s",s) != EOF)
    {
        len = strlen(s);
```

```
        l = 0;
        a[l++] = '.';
        a[l++] = ',';
        for (int i = 0;i < len;i++)
        {
            a[l++] = s[i];
            a[l++] = ',';
        }
        pnow = 0;
        res = 0;
        for (int i = 1;i < l;i++)
        {
            if (pnow > i)
                p[i] = min(p[2*pid-i],pnow-i);
            else
                p[i] = 1;
            for (;a[i-p[i]] == a[i+p[i]];p[i]++);
            if (i+p[i] > pnow)
            {
                pnow = i+p[i];
                pid = i;
            }
            if (p[i] > res)
            {
                res = p[i];
                resid = i;
            }
        }
        for (int i = resid-res+2;i < resid+res-1;i += 2)
            printf("%c",a[i]);
        printf("\n");
    }
    return 0;
}
```

3.7 *字符串最小表示法

```
int Gao(char a[],int len)
{
  int i = 0,j = 1,k = 0;
  while (i < len && j < len && k < len)
  {
    int cmp = a[(j+k)%len]-a[(i+k)%len];
    if (cmp == 0)
      k++;
    else
    {
      if (cmp > 0)
```

```
12          j += k+1;
13        else
14          i += k+1;
15        if (i == j) j++;
16        k = 0;
17      }
18    }
19    return min(i,j);
20  }
```

3.8 带*通配符的匹配

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  char a[110],b[110],sp[110][110],tot,place[110];
8  int n,la,lb,ll;
9
10 bool check(int id,int pos)
11 {
12     for (int i = 0;sp[id][i] != 0;i++)
13         if (b[pos+i] != sp[id][i])
14             return false;
15     return true;
16 }
17
18 bool check()
19 {
20     lb = strlen(b);
21     int pre = 0;
22     for (int i = 0;i < tot;i++)
23     {
24         bool find = false;
25         for (int j = pre;j < lb;j++)
26             if (check(i,j) == true)
27             {
28                 place[i] = j;
29                 pre = place[i]+1;
30                 find = true;
31                 break;
32             }
33         if (find == false)  return false;
34     }
35     if (a[0] != '*')
36         if (place[0] != 0)
37             return false;
```

```
38      if (a[la-1] != '*')
39          if (check(tot-1,lb-ll) == false)
40              return false;
41      return true;
42  }
43
44  int main()
45  {
46      while (scanf("%s",a) != EOF)
47      {
48          tot = 0;
49          for (int i = 0;a[i] != 0;i++)
50              if (a[i] != '*')
51              {
52                  int j;
53                  for (j = i;a[j] != 0 && a[j] != '*';j++)
54                      sp[tot][j-i] = a[j];
55                  sp[tot++][j-i] = 0;
56                  i = j;
57              }
58          la = strlen(a);
59          ll = strlen(sp[tot-1]);
60          scanf("%d",&n);
61          for (int i = 0;i < n;i++)
62          {
63              scanf("%s",b);
64              if (check() == true)
65                  puts(b);
66          }
67      }
68      return 0;
69  }
70  /*
71  Sample Input 1
72  *.*
73  4
74  main.c
75  a.out
76  readme
77  yacc
78
79  Sample Input 2
80  *a*a*a
81  4
82  aaa
83  aaaaa
84  aaaaax
85  abababa
```

```
86
87 Sample Output 1
88 main.c
89 a.out
90
91 Sample Output 2
92 aaa
93 aaaaa
94 abababa
95 */
```

4 数学

4.1 扩展GCD

求ax+by=gcd(a,b)的一组解

```
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if (b)
    {
        long long ret = ex_gcd(b,a%b,x,y),tmp = x;
        x = y;
        y = tmp-(a/b)*y;
        return ret;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
```

4.2 模线性方程组

```
//有更新
int m[10],a[10];//模数m 余数a
bool solve(int &m0,int &a0,int m,int a)//模线性方程组
{
    int y,x;
    int g=ex_gcd(m0,m,x,y);
    if (abs(a-a0)%g) return 0;
    x*=(a-a0)/g;
    x%=m/g;
    a0=(x*m0+a0);
    m0*=m/g;
    a0%=m0;
    if (a0<0) a0+=m0;
    return 1;
}
int MLES()
{
    bool flag=1;
    int m0=1,a0=0;
    for (int i=0; i<n; i++)
        if (!solve(m0,a0,m[i],a[i]))
        {
```

```
            flag=0;
            break;
        }
    if (flag)
        return a0;
    else
        return -1;
}
```

4.3 矩阵

乘法的时候将B数组转置一下然后$C[i][j] = \sum A[i][k] \times B[j][k]$会有奇效。

```
struct Matrix
{
    int a[52][52];
    Matrix operator * (const Matrix &b)const
    {
        Matrix res;
        for (int i = 0; i < 52; i++)
            for (int j = 0; j < 52; j++)
            {
                res.a[i][j] = 0;
                for (int k = 0; k < 52; k++)
                    res.a[i][j] += a[i][k] * b.a[k][j];
            }
        return res;
    }
    Matrix operator ^ (int y)const
    {
        Matrix res, x;
        for (int i = 0; i < 52; i++)
        {
            for (int j = 0; j < 52; j++)
                res.a[i][j] = 0, x.a[i][j] = a[i][j];
            res.a[i][i] = 1;
        }
        for (; y; y >>= 1, x = x * x)
            if (y & 1)
                res = res * x;
        return res;
    }
};
```

4.4 康拓展开

```
const int PermSize = 12;
int factory[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320,
   362880, 3628800, 39916800};
```

```
int Cantor(int a[])
{
    int i, j, counted;
    int result = 0;
    for (i = 0; i < PermSize; ++i)
    {
        counted = 0;
        for (j = i + 1; j < PermSize; ++j)
            if (a[i] > a[j])
                ++counted;
        result = result + counted * factory[PermSize - i - 1];
    }
    return result;
}

bool h[13];

void UnCantor(int x, int res[])
{
    int i,j,l,t;
    for (i = 1;i <= 12;i++)
        h[i] = false;
    for (i = 1; i <= 12; i++)
    {
        t = x / factory[12 - i];
        x -= t * factory[12 - i];
        for (j = 1, l = 0; l <= t; j++)
            if (!h[j])l++;
        j--;
        h[j] = true;
        res[i - 1] = j;
    }
}
```

4.5 FFT

```
const double PI= acos(-1.0);
struct vir
{
  double re,im; //实部和虚部
  vir(double a=0,double b=0)
  {
    re=a;
    im=b;
  }
  vir operator +(const vir &b)
  {return vir(re+b.re,im+b.im);}
  vir operator -(const vir &b)
  {return vir(re-b.re, im-b.im);}
```

```
14    vir operator *(const vir &b)
15    {return vir(re*b.re-im*b.im , re*b.im+im*b.re);}
16  };
17  vir x1[200005],x2[200005];
18  void change(vir *x,int len,int loglen)
19  {
20    int i,j,k,t;
21    for(i=0;i<len;i++)
22    {
23      t=i;
24      for(j=k=0; j<loglen; j++,t>>=1)
25        k= (k<<1)|(t&1);
26      if(k<i)
27      {
28      //  printf("%d %d\n",k,i);
29        vir wt=x[k];
30        x[k]=x[i];
31        x[i]=wt;
32      }
33    }
34  }
35  void fft(vir *x,int len,int loglen)
36  {
37    int i,j,t,s,e;
38    change(x,len,loglen);
39    t=1;
40    for(i=0;i<loglen;i++,t<<=1)
41    {
42      s=0;
43      e=s+t;
44      while(s<len)
45      {
46        vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
47        for(j=s;j<s+t;j++)
48        {
49          a=x[j];
50          b=x[j+t]*wn;
51          x[j]=a+b;
52          x[j+t]=a-b;
53          wn=wn*wo;
54        }
55        s=e+t;
56        e=s+t;
57      }
58    }
59  }
60  void dit_fft(vir *x,int len,int loglen)
61  {
```

```
62    int i,j,s,e,t=1<<loglen;
63    for(i=0;i<loglen;i++)
64    {
65      t>>=1;
66      s=0;
67      e=s+t;
68      while(s<len)
69      {
70        vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
71        for(j=s;j<s+t;j++)
72        {
73          a=x[j]+x[j+t];
74          b=(x[j]-x[j+t])*wn;
75          x[j]=a;
76          x[j+t]=b;
77          wn=wn*wo;
78        }
79        s=e+t;
80        e=s+t;
81      }
82    }
83    change(x,len,loglen);
84    for(i=0;i<len;i++)
85      x[i].re/=len;
86  }
87  int main()
88  {
89    char a[100005],b[100005];
90    int i,len1,len2,len,loglen;
91    int t,over;
92    while(scanf("%s%s",a,b)!=EOF)
93    {
94      len1=strlen(a)<<1;
95      len2=strlen(b)<<1;
96      len=1;loglen=0;
97      while(len<len1)
98      {
99        len<<=1;  loglen++;
100     }
101     while(len<len2)
102     {
103       len<<=1;  loglen++;
104     }
105     for(i=0;a[i];i++)
106     {
107       x1[i].re=a[i]-'0';
108       x1[i].im=0;
109     }
```

```
110     for(;i<len;i++)
111       x1[i].re=x1[i].im=0;
112     for(i=0;b[i];i++)
113     {
114       x2[i].re=b[i]-'0';
115       x2[i].im=0;
116     }
117     for(;i<len;i++)
118       x2[i].re=x2[i].im=0;
119     fft(x1,len,loglen);
120     fft(x2,len,loglen);
121     for(i=0;i<len;i++)
122       x1[i] = x1[i]*x2[i];
123     dit_fft(x1,len,loglen);
124     for(i=(len1+len2)/2-2,over=len=0;i>=0;i--)
125     {
126       t=(int)(x1[i].re+over+0.5);
127       a[len++]= t%10;
128       over = t/10;
129     }
130     while(over)
131     {
132       a[len++]=over%10;
133       over/=10;
134     }
135     for(len--;len>=0&&!a[len];len--);
136       if(len<0)
137       putchar('0');
138       else
139         for(;len>=0;len--)
140           putchar(a[len]+'0');
141     putchar('\n');
142   }
143   return 0;
144 }
```

4.6 爬山法计算器

注意灵活运用。
双目运算符在calc()中，左结合单目运算符在P()中，右结合单目运算符在calc_exp中。（但是还没遇到过。。）

```
1 #include <iostream>
2 #include <cstdio>
3 #include <cstring>
4 #include <algorithm>
5 #include <string>
6 using namespace std;
7
```

```
8  char s[100000];
9  int n,cur;
10 const string OP = "+-*";
11
12 char next_char()
13 {
14     if (cur >= n) return EOF;
15     return s[cur];
16 }
17
18 int get_priority(char ch)
19 {
20     if (ch == '*')  return 2;
21     return 1;
22 }
23
24 int P();
25
26 int calc(int a,char op,int b)
27 {
28     if (op == '+')
29         return a+b;
30     if (op == '-')
31         return a-b;
32     if (op == '*')
33         return a*b;
34 }
35
36 int calc_exp(int p)
37 {
38     int a = P();
39     while ((OP.find(next_char()) != OP.npos) && (get_priority(
          next_char()) >= p))
40     {
41         char op = next_char();
42         cur++;
43         a = calc(a,op,calc_exp(get_priority(op)+1));
44     }
45     return a;
46 }
47
48 int totvar,m,var[26],varid[26];
49
50 int P()
51 {
52     if (next_char() == '-')
53     {
54         cur++;
```

```
55          return -P();
56      }
57      else if (next_char() == '+')
58      {
59          cur++;
60          return P();
61      }
62      else if (next_char() == '(')
63      {
64          cur++;
65          int res = calc_exp(0);
66          cur++;
67          return res;
68      }
69      else
70      {
71          cur++;
72          //cout << "getvar at " << cur << ' ' << var[varid[s[cur
               ]-'a']] << endl;
73          return var[varid[s[cur-1]-'a']];
74      }
75  }
76
77  int id[26],minid;
78
79  int main()
80  {
81      while (true)
82      {
83          scanf("%d%d",&totvar,&var[0]);
84          if (totvar == 0 && var[0] == 0)  break;
85          for (int i = 1;i < totvar;i++)
86              scanf("%d",&var[i]);
87          scanf("%d",&m);
88          scanf("%s",s);
89          for (int i = 0;i < 26;i++)
90              id[i] = -1;
91          minid = 0;
92          n = strlen(s);
93          for (int i = 0;i < n;i++)
94              if (s[i] >= 'a' && s[i] <= 'z')
95              {
96                  if (id[s[i]-'a'] == -1)
97                  {
98                      id[s[i]-'a'] = minid;
99                      minid++;
100                 }
101                 s[i] = 'a'+id[s[i]-'a'];
```

```
102             }
103         for (int i = 0;i < totvar;i++)
104             varid[i] = i;
105         int res = 0;
106         do
107         {
108             cur = 0;
109             int tmp = calc_exp(0);
110             if (tmp == m)
111             {
112                 res++;
113                 break;
114             }
115         }
116         while (next_permutation(varid,varid+totvar));
117         //puts(s);
118         if (res > 0)
119             puts("YES");
120         else
121             puts("NO");
122     }
123   return 0;
124 }
```

4.7 线性筛

```
1  int N;
2  bool isPrime[10001];
3  int prime[10000];
4  void getPrime(int n)
5  {
6      memset(isPrime,1,++n);
7      N=0;
8      isPrime[0]=isPrime[1]=0;
9      for (int i=2;i<n;i++)
10     {
11         if (isPrime[i])
12             prime[N++]=i;
13         for (int j=0;j<N && prime[j]*i<n;j++)
14         {
15             isPrime[i*prime[j]]=0;
16             if (i%prime[j]==0)
17                 break;
18         }
19     }
20 }
```

4.8 线性规划

```
#define MAXM 20  //max num of basic varibles
#define INF 1E200

double A[MAXM+5][MAXN+MAXM+5];
double b[MAXM+5],c[MAXN+MAXM+5];
int N[MAXN+5],B[MAXM+5];
double X[MAXN+MAXM+5],V;
int n,m,R,C,nCnt,bCnt;
int v1[MAXN],v2[MAXN];

int fcmp(double a,double b)
{
  if(fabs(a-b)<1E-7) return 0;
  if(a>b) return 1;
  return -1;
}

void Pivot(int l,int e)
{
  double t=A[l][e],p=c[e];
  b[l]=b[l]/t;
  for(int i=1;i<=C;i++)
    A[l][i]/=t;
  V=V-c[e]*b[l];
  for(int i=1;i<=R;i++)
  {
    if(i==l||fcmp(A[i][e],0.0)==0)
      continue;
    t=A[i][e];
    b[i]=b[i]-t*b[l];
    for(int j=1;j<=C;j++)
      A[i][j]=A[i][j]-t*A[l][j];
  }
  for(int i=1;i<=C;i++)
    c[i]=c[i]-p*A[l][i];
  for(int i=1;i<=nCnt;i++)
  {
    if(N[i]==e)
    {
      N[i]=B[l];
      break;
    }
  }
  B[l]=e;
}

bool Process(double P[])
{
```

```
49    while(true)
50    {
51      int e=-1;
52      double mV=-INF;
53      for(int i=1;i<=nCnt;i++)
54        if(fcmp(P[N[i]],mV)==1)
55          mV=P[N[i]],e=N[i];
56
57      if(fcmp(mV,0.0)<=0) break;
58      int l=-1;
59      mV=INF;
60      for(int i=1;i<=bCnt;i++)
61      {
62        if(fcmp(A[i][e],0.0)==1)
63        {
64          double t=b[i]/A[i][e];
65          if(fcmp(mV,t)==1||(fcmp(mV,t)==0&&(l==-1||B[l]>B[i])))
66            mV=t,l=i;
67        }
68      }
69      if(l==-1) return false;
70      Pivot(l,e);
71    }
72    return true;
73  }
74
75  bool initSimplex()
76  {
77    nCnt=bCnt=0;
78    for(int i=1;i<=n;i++)
79      N[++nCnt]=i;
80    for(int i=1;i<=m;i++)
81      B[++bCnt]=i+n,A[i][n+i]=1.0;
82    R=bCnt,C=bCnt+nCnt;
83    double minV=INF;
84    int p=-1;
85    for(int i=1;i<=m;i++)
86      if(fcmp(minV,b[i])==1)
87        minV=b[i],p=i;
88    if(fcmp(minV,0.0)>=0)
89      return true;
90    N[++nCnt]=n+m+1;R++,C++;
91    for(int i=0;i<=C;i++)
92      A[R][i]=0.0;
93    for(int i=1;i<=R;i++)
94      A[i][n+m+1]=-1.0;
95    Pivot(p,n+m+1);
96    if(!Process(A[R])) return false;
```

```
97    if(fcmp(b[R],0.0)!=0)
98      return false;
99    p=-1;
100   for(int i=1;i<=bCnt&&p==-1;i++)
101     if(B[i]==n+m+1) p=i;
102   if(p!=-1)
103   {
104     for(int i=1;i<=nCnt;i++)
105     {
106       if(fcmp(A[p][N[i]],0.0)!=0)
107       {
108         Pivot(p,N[i]);
109         break;
110       }
111     }
112   }
113   bool f=false;
114   for(int i=1;i<=nCnt;i++)
115   {
116     if(N[i]==n+m+1) f=true;
117     if(f&&i+1<=nCnt)
118       N[i]=N[i+1];
119   }
120   nCnt--;
121   R--,C--;
122   return true;
123 }
124
125 //-1: no solution 1: no bound 0: has a solution -V
126 int Simplex()
127 {
128   if(!initSimplex())
129     return -1;
130   if(!Process(c))
131     return 1;
132   for(int i=1;i<=nCnt;i++)
133     X[N[i]]=0.0;
134   for(int i=1;i<=bCnt;i++)
135     X[B[i]]=b[i];
136   return 0;
137 }
138
139 int main()
140 {
141     //n = 1;m=1;
142     //V= 0.0;
143     //c[1] = 1.0;
144     //A[1][1] = 1.0;
```

```
145     //b[1] = 5.0;
146     //Simplex();
147     //printf("V = %.3f\n",V);
148
149   while(scanf("%d",&v1[1]) == 1)
150         {
151             for(int i = 2; i<=6;i++)
152                 scanf("%d",&v1[i]);
153             n = 4; m = 6;
154             for(int i = 0 ; i<=m+1;i++)
155                 for(int j=0;j<=n+m+2;j++)
156                     A[i][j] = c[j] = 0;
157             memset(b,0,sizeof(b));
158             V = 0.0;
159             /*
160             n 为未知数个数
161             m 为约束个数
162             目标: siama(c[i]*xi)
163             约束: sigma(A[i][j]*xj) <=b[i]; j = 1 ...  n
164             解存在X里面
165             */
166             b[1] = v1[1] ; A[1][1] = 1;A[1][4] = 1;
167             b[2] = v1[2] ; A[2][1] = 1;A[2][3] = 1;
168             b[3] = v1[3] ; A[3][3] = 1;A[3][4] = 1;
169             b[4] = v1[4] ; A[4][2] = 1;A[4][3] = 1;
170             b[5] = v1[5] ; A[5][2] = 1;A[5][4] = 1;
171             b[6] = v1[6] ; A[6][1] = 1;A[6][2] = 1;
172             c[1] = 1;c[2] = 1;c[3] = 1;c[4] = 1;
173             Simplex();
174             //printf("V = %.3f\n",V);
175             printf("%.3f␣%.3f␣%.3f␣%.3f\n",X[1],X[2],X[3],X[4]);
176
177         }
178   return 0;
179 }
```

4.9 分解质因数

4.9.1 米勒拉宾+分解因数

```
1 #include<ctime>
2 #include<iostream>
3 #define bint long long
4 using namespace std;
5 const int TIME = 8;//测试次数，够了8~10
6 int factor[100],fac_top = -1;
7
8 //计算两个数的gcd
9 bint gcd(bint small,bint big)
```

```
10 {
11     while(small)
12     {
13         swap(small,big);
14         small%=big;
15     }
16     return abs(big);
17 }
18
19 //ret = (a*b)%n (n<2^62)
20 bint muti_mod(bint a,bint b,bint n)
21 {
22     bint exp = a%n, res = 0;
23     while(b)
24     {
25         if(b&1)
26         {
27             res += exp;
28             if(res>n) res -= n;
29         }
30         exp <<= 1;
31         if (exp>n) exp -= n;
32         b>>=1;
33     }
34     return res;
35 }
36
37 // ret = (a^b)%n
38 bint mod_exp(bint a,bint p,bint m)
39 {
40     bint exp=a%m, res=1; //
41     while(p>1)
42     {
43         if(p&1)
44             res=muti_mod(res,exp,m);
45         exp = muti_mod(exp,exp,m);
46         p>>=1;
47     }
48     return muti_mod(res,exp,m);
49 }
50
51 //miller-法测试素数rabin, time 测试次数
52 bool miller_rabin(bint n, int times)
53 {
54     if(n==2)return 1;
55     if(n<2||!(n&1))return 0;
56     bint a, u=n-1, x, y;
57     int t=0;
```

```
58      while(u%2==0)
59      {
60          t++;
61          u/=2;
62      }
63      srand(time(0));
64      for(int i=0; i<times; i++)
65      {
66          a = rand() % (n-1) + 1;
67          x = mod_exp(a, u, n);
68          for(int j=0; j<t; j++)
69          {
70              y = muti_mod(x, x, n);
71              if ( y == 1 && x != 1 && x != n-1 )
72                  return false; //must not
73              x = y;
74          }
75          if( y!=1) return false;
76      }
77      return true;
78  }
79
80  bint pollard_rho(bint n,int c)//找出一个因子
81  {
82      bint x,y,d,i = 1,k = 2;
83      srand(time(0));
84      x = rand()%(n-1)+1;
85      y = x;
86      while(true)
87      {
88          i++;
89          x = (muti_mod(x,x,n) + c) % n;
90          d = gcd(y-x, n);
91          if (1 < d && d < n) return d;
92          if( y == x) return n;
93          if(i == k)
94          {
95              y = x;
96              k <<= 1;
97          }
98      }
99  }
100
101 void findFactor(bint n,int k)//二分找出所有质因子，存入factor
102 {
103     if(n==1)return;
104     if(miller_rabin(n, TIME))
105     {
```

```
106         factor[++fac_top] = n;
107         return;
108     }
109     bint p = n;
110     while(p >= n)
111         p = pollard_rho(p,k--);//值变化，防止死循环k
112     findFactor(p,k);
113     findFactor(n/p,k);
114 }
115
116 int main()
117 {
118     bint cs,n,min;
119     cin>>cs;
120     while (cs--)
121     {
122         cin>>n;
123         fac_top = min = -1;
124         if(miller_rabin(n,TIME)) cout<<"Prime"<<endl;
125         else
126         {
127             findFactor(n,107);
128             for(int i=0; i<=fac_top; i++)
129             {
130                 if(min<0||factor[i]<min)
131                     min = factor[i];
132             }
133             cout<<min<<endl;
134         }
135     }
136     return 0;
137 }
```

4.9.2 暴力版本

```
1  int N;
2  int num[30],fac[30];
3  void getFactor(int x)
4  {
5      N=0;
6      memset(num,0,sizeof(num));
7      for (int i=0; prime[i]*prime[i]<=x && i<L; i++)
8      {
9          if (x%prime[i]==0)
10         {
11             while (x%prime[i]==0)
12             {
13                 x/=prime[i];
14                 num[N]++;
```

```
15              }
16              fac[N++]=prime[i];
17          }
18      }
19      if (x>1)
20      {
21          num[N]=1;
22          fac[N++]=x;
23      }
24  }
```

4.10 baby step giant step

```
1  #define MOD 76543
2  int hs[MOD], head[MOD], next[MOD], id[MOD], top;
3  void insert(int x, int y)
4  {
5      int k = x%MOD;
6      hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top
           ++;
7  }
8  int find(int x)
9  {
10     int k = x%MOD;
11     for (int i = head[k]; i; i = next[i]) if (hs[i] == x) return
           id[i];
12     return -1;
13 }
14 int BSGS(int a, int b, int n)
15 {
16     memset(head, 0, sizeof(head));
17     top = 1;
18     if (b==1) return 0;
19     int m = sqrt(n+.0), j;
20     long long x = 1, p = 1;
21     for (int i = 0; i < m; ++i, p = p*a%n) insert(p*b%n, i);
22     for (long long i = m; ; i += m)
23     {
24         if ((j = find(x=x*p%n)) != -1) return i-j;
25         if (i > n) break;
26     }
27     return -1;
28 }
```

4.11 原根

```
1  int getPriRoot(int p)
2  {
3      if (p==2) return 1;
```

```
    int phi = p - 1;
    getFactor(phi);
    for (int g = 2; g < p; ++g)
    {
        bool flag=1;
        for (int i = 0; flag && i < N; ++i)
            if (power(g, phi/fac[i], p) == 1)
                flag=0;
        if (flag)
            return g;
    }
}
```

4.12 逆元

```
void getInv2(int x)
{
    inv[1]=1;
    for (int i=2; i<=x; i++)
        inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
}
int getInv(int x)//为素数mod
{
    return power(x,mod-2);
}
```

4.13 卢卡斯

卢卡斯，$num[i]$阶乘也

```
int comLucus(int n,int m,int p)
{
    int ans=1;
    for (; n && m && ans; n/=p,m/=p)
    {
        if (n%p>=m%p)
            ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p*getInv(num[
               n%p-m%p])%p;
        else
            ans=0;
    }
    return ans;
}
```

4.14 欧拉函数

4.14.1 分解质因数

```
int getEuler(int x)
{
```

```
3       getFactor(x);
4       int ret=x;
5       for (int i=0; i<N; i++)
6           ret = ret/fac[i]*(fac[i]-1);
7       return ret;
8   }
```

4.14.2 一次预处理

```
1   void getEuler2()
2   {
3       memset(euler,0,sizeof(euler));
4       euler[1] = 1;
5       for (int i = 2; i <= 3000000; i++)
6       {
7           if (!euler[i])
8           {
9               for (int j = i; j <= 3000000; j += i)
10              {
11                  if (!euler[j])
12                      euler[j] = j;
13                  euler[j] = euler[j]/i*(i-1);
14              }
15          }
16      }
17  }
```

4.15 费马降阶法

分解素数p为x^2+y^2的费马降阶法，失败返回-1，主程序调用calcu(p,x,y)

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <stdlib.h>
4   int p,expp,A,B,aa,ans,tt;
5   long long M;
6   long long exp(int a,int b,long long mod)
7   {
8       long long ans=1,num=a;
9       while (b!=0)
10      {
11          if (b&1)
12          {
13              ans=((ans%mod)*(num%mod))%mod;
14          }
15          num=((num%mod)*(num%mod))%mod;
16          b>>=1;
17      }
18      return ans;
19  }
20  int calcu(int p,int &x,int &y)
```

```
{
        if (p%4!=1) return -1;
        else
        {
            expp=(p-1)/4;
            A,B;
            while (1)
            {
                aa=rand()%p;
                if (aa==0) continue;
                A=exp(aa,expp,p);
                ans=(((long long)A%p)*((long long)A%p))%p;
                if (ans==p-1) break;
            }
            B=1;
            M=((long long)A*(long long)A+(long long)B*(long
               long)B)/p;
            if (M!=1) B=p;
            while (M!=1)
            {
                  if (B>A)
                  {tt=A; A=B; B=tt;}
                  tt=A;
                  A=B;
                  B=tt%B;
                  M=((long long)A*(long long)A+(long long)B*(
                     long long)B)/p;
            }
            if (B<=A)
            {
                     x=B;
                     y=A;
            }
            else
            {
               x=A;
               y=B;
            }
        }
}
int main()
{
    while (scanf("%d",&p)!=EOF)
    {
          int x,y;
          if (calcu(p,x,y)!=-1)
    }
    return 0;
```

```
67 }
```

4.16 自适应simp

过了哈尔滨积分题，精度要求不高的时候可以考虑使用。
暂时我只能用这个做做类似于凸函数或者凹函数的函数。

```
1  double Simp(double l,double r)
2  {
3      double h = (r-l)/2.0;
4      return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
5  }
6
7  double rSimp(double l,double r)
8  {
9      double mid = (l+r)/2.0;
10     if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
11         return Simp(l,r);
12     else
13         return rSimp(l,mid)+rSimp(mid,r);
14 }
```

4.17 组合数求模

模是质数

```
1  #include<cstdio>
2  #include<cstring>
3  #include<iostream>
4  using namespace std;
5  int mod;
6  long long num[100000];
7  int ni[100],mi[100];
8  int len;
9  void init(int p)
10 {
11     mod=p;
12     num[0]=1;
13     for (int i=1; i<p; i++)
14         num[i]=i*num[i-1]%p;
15 }
16 void get(int n,int ni[],int p)
17 {
18     for (int i = 0; i < 100; i++)
19         ni[i] = 0;
20     int tlen = 0;
21     while (n != 0)
22     {
23         ni[tlen++] = n%p;
```

```
24          n /= p;
25      }
26      len = tlen;
27  }
28  long long power(long long x,long long y)
29  {
30      long long ret=1;
31      for (long long a=x%mod; y; y>>=1,a=a*a%mod)
32          if (y&1)
33              ret=ret*a%mod;
34      return ret;
35  }
36  long long getInv(long long x)//mod为素数
37  {
38      return power(x,mod-2);
39  }
40  long long calc(int n,int m,int p)//C(n,m)%p
41  {
42      init(p);
43      long long ans=1;
44      for (; n && m && ans; n/=p,m/=p)
45      {
46          if (n%p>=m%p)
47              ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p*getInv(num[
                    n%p-m%p])%p;
48          else
49              ans=0;
50      }
51      return ans;
52  }
53  int main()
54  {
55      int t;
56      scanf("%d",&t);
57      while (t--)
58      {
59          int n,m,p;
60          scanf("%d%d%d",&n,&m,&p);
61          printf("%I64d\n",calc(n+m,m,p));
62      }
63      return 0;
64  }
```

4.18 其它公式

4.18.1 拉格朗日插值法

已知$y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$曲线上的$n$个点$(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$用拉格朗日插值法可以不求系数可知任意$x$对应的$y$值。

$$\begin{aligned} y = & y_1 \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} \\ & + y_2 \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} \\ & + \cdots \\ & + y_n \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} \end{aligned}$$

特别的，如果$x_1 \sim x_n$为 连续自然数，那么对于下一个自然数对应的y值为：

$$y_{n+1} = (-1)^{n-1}C_n^0 y_1 + (-1)^{n-2}C_n^1 y_2 + \cdots + (-1)^0 C_n^{n-1} y_n$$

这个组合系数可以通过高斯消元暴出来，前提是要猜到它满足递推关系。

4.18.2 正多面体顶点着色

正四面体：$N = \frac{(n^4+11\times n^2)}{12}$
正六面体：$N = \frac{(n^8+17\times n^4+6\times n^2)}{24}$
正八面体：$N = \frac{(n^6+3\times n^4+12\times n^3+8\times n^2)}{24}$
正十二面体：$N = \frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{60}$
正二十面体：$N = \frac{(n^{12}+15\times n^6+44\times n^4)}{60}$

4.18.3 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2 \times (2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$
$\sum k\times(k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k\times(k+1)\times(k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k\times(k+1)\times(k+2)\times(k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$

4.18.4 几何公式

球扇形:
全面积：$T = \pi r(2h + r_0)$，h为球冠高，r_0为球冠底面半径
体积：$V = \frac{2\pi r^2 h}{3}$

4.18.5 小公式

Pick 公式：$A = E \times 0.5 + I - 1$（A是多边形面积，E是边界上的整点，I是多边形内部的整点）

海伦公式：$S = \sqrt{p(p-a)(p-b)(p-c)}$，其中$p = \frac{(a+b+c)}{2}$，$abc$为三角形的三条边长

求$\binom{n}{k}$中素因子P的个数:

1. 把n转化为P进制，并记它每个位上的和为$S1$
2. 把$n-k$，k做同样的处理，得到$S2$，$S3$

则$\binom{n}{k}$中素因子P的个数：$\frac{S2+S3-S1}{P-1}$

部分错排公式:
$n+m$个数中m个数必须错排 求排列数

```
dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
dp[0] = n!;
dp[1] = n*n!;
```

$dp[m]$为所求解

5 数据结构

5.1 *Splay

持续学习中。
注意节点的size值不一定是真实的值！如果有需要需要特别维护！

1. 旋转和Splay操作
2. rank操作
3. insert操作（。。很多题目都有）
4. del操作（郁闷的出纳员）
5. 由数组建立Splay
6. 前驱后继（营业额统计）
7. Pushdown Pushup的位置
8. *。。。暂时想不起了

节点定义。。

```
const int MaxN = 50003;

struct Node
{
    int size,key;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil;
```

无内存池的几个初始化函数。

```
Node *newNode(int v, Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur++;
}

void Init()
{
    cur = mem;
    nil = newNode(0, cur);
    nil->size = 0;
}
```

带内存池的几个函数。

```
int emp[MaxN], totemp;

Node *newNode(int v, Node *p)
{
    cur = mem + emp[--totemp];
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur;
}

void Init()
{
    for (int i = 0; i < MaxN; ++i)
        emp[i] = i;
    totemp = MaxN;
    cur = mem + emp[--totemp];
    nil = newNode(0, cur);
    nil->size = 0;
}

void Recycle(Node *p)
{
    if (p == nil)   return;
    Recycle(p->c[0]), Recycle(p->c[1]);
    emp[totemp++] = p - mem;
}
```

基本的Splay框架。维护序列用。
一切下标从0开始。

```
struct SplayTree
{
    Node *root;
    void Init()
    {
        root = nil;
    }
    void Pushup(Node *x)
    {
        if (x == nil)   return;
        Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
        x->size = x->c[0]->size + x->c[1]->size + 1;
    }
    void Pushdown(Node *x)
    {
```

```
16          if (x == nil)   return;
17          //do something
18      }
19      void Rotate(Node *x, int f)
20      {
21          if (x == nil)   return;
22          Node *y = x->p;
23          y->c[f ^ 1] = x->c[f], x->p = y->p;
24          if (x->c[f] != nil)
25              x->c[f]->p = y;
26          if (y->p != nil)
27              y->p->c[y->p->c[1] == y] = x;
28          x->c[f] = y, y->p = x;
29          Pushup(y);
30      }
31      void Splay(Node *x, Node *f)
32      {
33          while (x->p != f)
34          {
35              Node *y = x->p;
36              if (y->p == f)
37                  Rotate(x, x == y->c[0]);
38              else
39              {
40                  int fd = y->p->c[0] == y;
41                  if (y->c[fd] == x)
42                      Rotate(x, fd ^ 1), Rotate(x, fd);
43                  else
44                      Rotate(y, fd), Rotate(x, fd);
45              }
46          }
47          Pushup(x);
48          if (f == nil)
49              root = x;
50      }
51      void Select(int k, Node *f)
52      {
53          Node *x = root;
54          Pushdown(x);
55          int tmp;
56          while ((tmp = x->c[0]->size) != k)
57          {
58              if (k < tmp)    x = x->c[0];
59              else
60                  x = x->c[1], k -= tmp + 1;
61              Pushdown(x);
62          }
63          Splay(x, f);
```

```
64      }
65      void Select(int l, int r)
66      {
67          Select(l, nil), Select(r + 2, root);
68      }
69      Node *Make_tree(int a[], int l, int r, Node *p)
70      {
71          if (l > r)  return nil;
72          int mid = l + r >> 1;
73          Node *x = newNode(a[mid], p);
74          x->c[0] = Make_tree(a, l, mid - 1, x);
75          x->c[1] = Make_tree(a, mid + 1, r, x);
76          Pushup(x);
77          return x;
78      }
79      void Insert(int pos, int a[], int n)
80      {
81          Select(pos, nil), Select(pos + 1, root);
82          root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
83          Splay(root->c[1]->c[0], nil);
84      }
85      void Insert(int v)
86      {
87          Node *x = root, *y = nil;
88          while (x != nil)
89          {
90              y = x;
91              y->size++;
92              x = x->c[v >= x->key];
93          }
94          y->c[v >= y->key] = x = newNode(v, y);
95          Splay(x, nil);
96      }
97      void Remove(int l, int r)
98      {
99          Select(l, r);
100         //Recycle(root->c[1]->c[0]);
101         root->c[1]->c[0] = nil;
102         Splay(root->c[1], nil);
103     }
104 };
```

例题：旋转区间赋值求和求最大子序列。
注意打上懒标记后立即Pushup。Pushup(root-c[1]-c[0]),Pushup(root-c[1]),Pushup(root);

```
1   void Pushup(Node *x)
2   {
3     if (x == nil) return;
4     Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
```

```
5       x->size = x->c[0]->size+x->c[1]->size+1;
6
7       x->sum = x->c[0]->sum+x->c[1]->sum+x->key;
8       x->lsum = max(x->c[0]->lsum,x->c[0]->sum+x->key+max(0,x->c
            [1]->lsum));
9       x->rsum = max(x->c[1]->rsum,x->c[1]->sum+x->key+max(0,x->c
            [0]->rsum));
10      x->maxsum = max(max(x->c[0]->maxsum,x->c[1]->maxsum),x->key+
            max(0,x->c[0]->rsum)+max(0,x->c[1]->lsum));
11    }
12    void Pushdown(Node *x)
13    {
14      if (x == nil) return;
15      if (x->rev)
16      {
17        x->rev = 0;
18        x->c[0]->rev ^= 1;
19        x->c[1]->rev ^= 1;
20        swap(x->c[0],x->c[1]);
21
22        swap(x->lsum,x->rsum);
23      }
24      if (x->same)
25      {
26        x->same = false;
27        x->key = x->lazy;
28        x->sum = x->key*x->size;
29        x->lsum = x->rsum = x->maxsum = max(x->key,x->sum);
30        x->c[0]->same = true, x->c[0]->lazy = x->key;
31        x->c[1]->same = true, x->c[1]->lazy = x->key;
32      }
33    }
34
35  int main()
36  {
37    int totcas;
38    scanf("%d",&totcas);
39    for (int cas = 1;cas <= totcas;cas++)
40    {
41      Init();
42      sp.Init();
43      nil->lsum = nil->rsum = nil->maxsum = -Inf;
44      sp.Insert(0);
45      sp.Insert(0);
46
47      int n,m;
48      scanf("%d%d",&n,&m);
49      for (int i = 0;i < n;i++)
```

```
50        scanf("%d",&a[i]);
51      sp.Insert(0,a,n);
52
53      for (int i = 0;i < m;i++)
54      {
55        int pos,tot,c;
56        scanf("%s",buf);
57        if (strcmp(buf,"MAKE-SAME") == 0)
58        {
59          scanf("%d%d%d",&pos,&tot,&c);
60          sp.Select(pos-1,pos+tot-2);
61          sp.root->c[1]->c[0]->same = true;
62          sp.root->c[1]->c[0]->lazy = c;
63          sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
64        }
65        else if (strcmp(buf,"INSERT") == 0)
66        {
67          scanf("%d%d",&pos,&tot);
68          for (int i = 0;i < tot;i++)
69            scanf("%d",&a[i]);
70          sp.Insert(pos,a,tot);
71        }
72        else if (strcmp(buf,"DELETE") == 0)
73        {
74          scanf("%d%d",&pos,&tot);
75          sp.Remove(pos-1,pos+tot-2);
76        }
77        else if (strcmp(buf,"REVERSE") == 0)
78        {
79          scanf("%d%d",&pos,&tot);
80          sp.Select(pos-1,pos+tot-2);
81          sp.root->c[1]->c[0]->rev ^= 1;
82          sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
83        }
84        else if (strcmp(buf,"GET-SUM") == 0)
85        {
86          scanf("%d%d",&pos,&tot);
87          sp.Select(pos-1,pos+tot-2);
88          printf("%d\n",sp.root->c[1]->c[0]->sum);
89        }
90        else if (strcmp(buf,"MAX-SUM") == 0)
91        {
92          sp.Select(0,sp.root->size-3);
93          printf("%d\n",sp.root->c[1]->c[0]->maxsum);
94        }
95      }
96    }
97    return 0;
```

```
98 }
```

维护多个序列的时候，不需要建立很多Splay。只需要记录某个点在内存池中的绝对位置就可以了。
需要操作它所在的序列时直接Splay到nil。此时Splay的root所在的Splay就是这个序列了。
新建序列的时候需要多加入两个额外节点。如果某个Splay只有两个节点了需要及时回收。
例题：Box（维护括号序列）

```
1   \\下面都是专用函数
2   \\判断x在不在f里面
3   bool Ancestor(Node *x,Node *f)
4   {
5       if (x == f) return true;
6       while (x->p != nil)
7       {
8           if (x->p == f)  return true;
9           x = x->p;
10      }
11      return false;
12  }
13  \\把Splay v插入到pos后面，pos=nil时新开一个序列
14  void Insert(Node *pos, Node *v)
15  {
16      int pl;
17      if (pos == nil)
18      {
19          Init();
20          Insert(0), Insert(0);
21          pl = 0;
22      }
23      else
24      {
25          Splay(pos, nil);
26          pl = root->c[0]->size;
27      }
28      Select(pl, nil), Select(pl + 1, root);
29      root->c[1]->c[0] = v;
30      v->p = root->c[1];
31      Splay(v, nil);
32  }
33  \\把[l,r]转出来（这里记录的是绝对位置）
34  void Select(Node *l, Node *r)
35  {
36  Splay(l, nil);
37      int pl = root->c[0]->size - 1;
38      Splay(r, nil);
39      int pr = root->c[0]->size - 1;
40      Select(pl, pr);
```

```
41      }
42      \\分离[l,r]
43      Node *Split(Node *l, Node *r)
44      {
45          Select(l, r);
46          Node *res = root->c[1]->c[0];
47          root->c[1]->c[0] = res->p = nil;
48          Splay(root->c[1], nil);
49          if (root->size == 2)
50          {
51              Recycle(root);
52              Init();
53          }
54          return res;
55      }
56
57  int main(int argc, char const *argv[])
58  {
59      freopen("P.in", "r", stdin);
60      bool first = true;
61      while (scanf("%d", &n) != EOF)
62      {
63          if (!first) puts("");
64          first = false;
65          Init();
66          for (int i = 0; i < n; i++)
67          {
68              \\建立独立的N个区间，记录绝对位置
69              sp.Init();
70              sp.Insert(0), sp.Insert(0);
71              sp.Insert(0,i+1),sp.Insert(1,i+1);
72              sp.Select(0, 0), l[i] = sp.root->c[1]->c[0];
73              sp.Select(1, 1), r[i] = sp.root->c[1]->c[0];
74          }
75          for (int i = 0; i < n; i++)
76          {
77              int f;
78              scanf("%d", &f);
79              if (f != 0)
80              {
81                  \\把[l[i],r[i]]插入到l[f-1]后面
82                  Node *pos = sp.Split(l[i], r[i]);
83                  sp.Insert(l[f - 1], pos);
84              }
85          }
86          scanf("%d", &n);
87          for (int i = 0; i < n; i++)
88          {
```

```
89              scanf("%s", com);
90              if (com[0] == 'Q')
91              {
92                  int pos;
93                  scanf("%d", &pos);
94                  \\求[l[pos-1],r[pos-1]]在哪个序列里面
95                  sp.Splay(l[pos - 1], nil);
96                  sp.Select(1, nil);
97                  printf("%d\n", sp.root->key);
98              }
99              else
100             {
101                 int u, v;
102                 scanf("%d%d", &u, &v);
103                 if (v == 0)
104                     sp.Insert(nil, sp.Split(l[u-1], r[u-1]));
105                 else
106                 {
107                     sp.Select(l[u-1],r[u-1]);
108                     if (sp.Ancestor(l[v-1],sp.root->c[1]->c[0])
                           == false)
109                       sp.Insert(l[v - 1], sp.Split(l[u-1], r[u
                             -1]));
110                 }
111             }
112         }
113     }
114     return 0;
115 }
```

5.2 动态树

懒标记是否及时Pushdown了?
修改之后有没有及时Pushup?

5.2.1 维护点权

查询链上的最长字段和
GetRoute是用换根写的

```
1  const int MaxN = 110000;
2
3  struct Node
4  {
5      int size, key;
6      bool rev;
7
```

```
8  //    bool same;
9  //    int lsum, rsum, sum, maxsum, sa;
10
11     Node *c[2];
12     Node *p;
13 } mem[MaxN], *cur, *nil, *pos[MaxN];
14
15 Node *newNode(int v, Node *p)
16 {
17     cur->c[0] = cur->c[1] = nil, cur->p = p;
18     cur->size = 1;
19     cur->key = v;
20     cur->rev = false;
21
22 //    cur->same = false;
23 //    cur->sa = 0;
24 //    cur->lsum = cur->rsum = cur->maxsum = 0;
25 //    cur->sum = v;
26
27     return cur++;
28 }
29
30 void Init()
31 {
32     cur = mem;
33     nil = newNode(0, cur);
34     nil->size = 0;
35 }
36
37 struct SplayTree
38 {
39     void Pushup(Node *x)
40     {
41         if (x == nil)   return;
42         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
43         x->size = x->c[0]->size + x->c[1]->size + 1;
44
45 //        x->sum = x->c[0]->sum + x->c[1]->sum + x->key;
46 //        x->lsum = max(x->c[0]->lsum, x->c[0]->sum + x->key +
       max(0, x->c[1]->lsum));
47 //        x->rsum = max(x->c[1]->rsum, x->c[1]->sum + x->key +
       max(0, x->c[0]->rsum));
48 //        x->maxsum = max(max(x->c[0]->maxsum, x->c[1]->maxsum),
49 //            x->key + max(0, x->c[0]->rsum) + max(0, x->c[1]->
       lsum));
50
51     }
52     void Pushdown(Node *x)
```

```
53      {
54          if (x == nil)   return;
55          if (x->rev)
56          {
57              x->rev = 0;
58              x->c[0]->rev ^= 1;
59              x->c[1]->rev ^= 1;
60              swap(x->c[0], x->c[1]);
61  //注意修改与位置有关的量
62  //              swap(x->lsum,x->rsum);
63          }
64
65  //          if (x->same)
66  //          {
67  //              x->same = false;
68  //              x->key = x->sa;
69  //              x->sum = x->sa * x->size;
70  //              x->lsum = x->rsum = x->maxsum = max(0, x->sum);
71  //              if (x->c[0] != nil)
72  //                  x->c[0]->same = true, x->c[0]->sa = x->sa;
73  //              if (x->c[1] != nil)
74  //                  x->c[1]->same = true, x->c[1]->sa = x->sa;
75  //          }
76      }
77      bool isRoot(Node *x)
78      {
79          return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x)
                ;
80      }
81      void Rotate(Node *x, int f)
82      {
83          if (isRoot(x))   return;
84          Node *y = x->p;
85          y->c[f ^ 1] = x->c[f], x->p = y->p;
86          if (x->c[f] != nil)
87              x->c[f]->p = y;
88          if (y != nil)
89          {
90              if (y == y->p->c[1])
91                  y->p->c[1] = x;
92              else if (y == y->p->c[0])
93                  y->p->c[0] = x;
94          }
95          x->c[f] = y, y->p = x;
96          Pushup(y);
97      }
98      void Splay(Node *x)
99      {
```

```
100         static Node *stack[MaxN];
101         int top = 0;
102         stack[top++] = x;
103         for (Node *y = x; !isRoot(y); y = y->p)
104             stack[top++] = y->p;
105         while (top)
106             Pushdown(stack[--top]);
107 
108         while (!isRoot(x))
109         {
110             Node *y = x->p;
111             if (isRoot(y))
112                 Rotate(x, x == y->c[0]);
113             else
114             {
115                 int fd = y->p->c[0] == y;
116                 if (y->c[fd] == x)
117                     Rotate(x, fd ^ 1), Rotate(x, fd);
118                 else
119                     Rotate(y, fd), Rotate(x, fd);
120             }
121         }
122         Pushup(x);
123     }
124     Node *Access(Node *u)
125     {
126         Node *v = nil;
127         while (u != nil)
128         {
129             Splay(u);
130             v->p = u;
131             u->c[1] = v;
132             Pushup(u);
133             u = (v = u)->p;
134             if (u == nil)
135                 return v;
136         }
137     }
138     Node *LCA(Node *u, Node *v)
139     {
140         Access(u);
141         return Access(v);
142     }
143     Node *Link(Node *u, Node *v)
144     {
145         Access(u);
146         Splay(u);
147         u->rev = true;
```

```
148         u->p = v;
149     }
150     void ChangeRoot(Node *u)
151     {
152         Access(u)->rev ^= 1;
153     }
154     Node *GetRoute(Node *u, Node *v)
155     {
156         ChangeRoot(u);
157         return Access(v);
158     }
159 };
160
161 int n, m;
162 SplayTree sp;
163
164 int main(int argc, char const *argv[])
165 {
166     while (scanf("%d", &n) != EOF)
167     {
168         Init();
169         for (int i = 0; i < n; i++)
170         {
171             int v;
172             scanf("%d", &v);
173             pos[i] = newNode(v, nil);
174         }
175         for (int i = 0; i < n - 1; i++)
176         {
177             int u, v;
178             scanf("%d%d", &u, &v);
179             u--, v--;
180             sp.Link(pos[u], pos[v]);
181         }
182
183 //         scanf("%d", &m);
184 //         for (int i = 0; i < m; i++)
185 //         {
186 //             int typ, u, v, c;
187 //             scanf("%d%d%d", &typ, &u, &v);
188 //             u--, v--;
189 //             if (typ == 1)
190 //                 printf("%d\n", sp.GetRoute(pos[u], pos[v])->
    maxsum);
191 //             else
192 //             {
193 //                 scanf("%d", &c);
194 //                 Node *p = sp.GetRoute(pos[u], pos[v]);
```

```
195 //                  p->same = true;
196 //                  p->sa = c;
197 //              }
198 //          }
199     }
200     return 0;
201 }
```

5.2.2 维护边权

刘汝佳的Happy Painting!
查询链上边的不同颜色数量
不能换根，但是可以Link和Cut

```
1  const int MaxN = 60000;
2
3  struct Node
4  {
5      int size,key;
6
7      int msk,lazy;
8
9      Node *c[2];
10     Node *p;
11 } mem[MaxN], *cur, *nil, *pos[MaxN];
12
13 Node *newNode(int v,Node *p)
14 {
15     cur->c[0] = cur->c[1] = nil, cur->p = p;
16     cur->size = 1;
17     cur->key = v;
18
19     cur->msk = 0;
20     cur->lazy = -1;
21
22     return cur++;
23 }
24
25 void Init()
26 {
27     cur = mem;
28     nil = newNode(0, cur);
29     nil->size = 0;
30 }
31
32 struct SplayTree
33 {
34     void Pushup(Node *x)
```

```
35      {
36          if (x == nil) return;
37          Pushdown(x);
38          Pushdown(x->c[0]);
39          Pushdown(x->c[1]);
40          x->size = x->c[0]->size + x->c[1]->size + 1;
41
42          x->msk = x->c[0]->msk | x->c[1]->msk | (1<<x->key);
43      }
44      void Pushdown(Node *x)
45      {
46          if (x == nil) return;
47
48          if (x->lazy != -1)
49          {
50              x->key = x->lazy;
51              x->msk = (1<<x->key);
52              x->c[0]->lazy = x->c[1]->lazy = x->lazy;
53              x->lazy = -1;
54          }
55      }
56      bool isRoot(Node *x)
57      {
58          return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x)
                ;
59      }
60      void Rotate(Node *x, int f)
61      {
62          if (isRoot(x))  return;
63          Node *y = x->p;
64          y->c[f ^ 1] = x->c[f], x->p = y->p;
65          if (x->c[f] != nil)
66              x->c[f]->p = y;
67          if (y != nil)
68          {
69              if (y == y->p->c[1])
70                  y->p->c[1] = x;
71              else if (y == y->p->c[0])
72                  y->p->c[0] = x;
73          }
74          x->c[f] = y, y->p = x;
75          Pushup(y);
76      }
77      void Splay(Node *x)
78      {
79          static Node *stack[MaxN];
80          int top = 0;
81          stack[top++] = x;
```

```
82          for (Node *y = x; !isRoot(y); y = y->p)
83              stack[top++] = y->p;
84          while (top)
85              Pushdown(stack[--top]);
86
87          while (!isRoot(x))
88          {
89              Node *y = x->p;
90              if (isRoot(y))
91                  Rotate(x, x == y->c[0]);
92              else
93              {
94                  int fd = y->p->c[0] == y;
95                  if (y->c[fd] == x)
96                      Rotate(x, fd ^ 1), Rotate(x, fd);
97                  else
98                      Rotate(y, fd), Rotate(x, fd);
99              }
100         }
101         Pushup(x);
102     }
103     Node *Access(Node *u)
104     {
105         Node *v = nil;
106         while (u != nil)
107         {
108             Splay(u);
109             v->p = u;
110             u->c[1] = v;
111             Pushup(u);
112             u = (v = u)->p;
113             if (u == nil) return v;
114         }
115     }
116     Node *Root(Node *u)
117     {
118         Access(u);
119         Splay(u);
120         for (Pushdown(u); u->c[0] != nil; u = u->c[0])
121             Pushdown(u);
122         Splay(u);
123         return u;
124     }
125     Node *LCA(Node *u, Node *v)
126     {
127         if (Root(u) != Root(v))
128             return nil;
129         Access(u);
```

```
130         return Access(v);
131     }
132     void Cut(Node *u)
133     {
134         Access(u);
135         Splay(u);
136         u->c[0] = u->c[0]->p = nil;
137         Pushup(u);
138     }
139     void Link(Node *u, Node *v, int val)
140     {
141         Access(u);
142         Splay(u);
143         u->p = v;
144         u->key = val;
145         Pushup(u);
146     }
147 };
148 
149 int cntbit(int x)
150 {
151     x = (x & 0x55555555) + ((x >> 1) & 0x55555555);
152     x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
153     x = (x & 0x0F0F0F0F) + ((x >> 4) & 0x0F0F0F0F);
154     x = (x & 0x00FF00FF) + ((x >> 8) & 0x00FF00FF);
155     x = (x & 0x0000FFFF) + ((x >> 16) & 0x0000FFFF);
156     return x;
157 }
158 
159 SplayTree sp;
160 int n,Q,f[MaxN];
161 
162 int main(int argc, char const *argv[])
163 {
164     while (scanf("%d%d",&n,&Q) != EOF)
165     {
166         Init();
167         for (int i = 0; i < n; i++)
168         {
169             scanf("%d",&f[i]);
170             pos[i] = newNode(0, nil);
171         }
172         for (int i = 0; i < n; i++)
173         {
174             int col;
175             scanf("%d",&col);
176             if (f[i] > 0)
177                 sp.Link(pos[i],pos[f[i]-1],col-1);
```

```
178         }
179         for (int q = 0; q < Q; q++)
180         {
181             int typ,x,y,c;
182             scanf("%d%d%d",&typ,&x,&y);
183             x--,y--;
184             if (typ == 3)
185             {
186                 Node *lca = sp.LCA(pos[x],pos[y]);
187                 if (lca == nil || x == y)
188                 {
189                     printf("0 0\n");
190                     continue;
191                 }
192                 int totedge = lca->c[1]->size;
193                 int msk = lca->c[1]->msk;
194 
195                 if (pos[x] != lca)
196                 {
197                     sp.Splay(pos[x]);
198                     totedge += pos[x]->size;
199                     msk |= pos[x]->msk;
200                 }
201 
202                 printf("%d %d\n",totedge,cntbit(msk));
203             }
204             else
205             {
206                 scanf("%d",&c);
207                 c--;
208                 if (typ == 1)
209                 {
210                     if (x == y) continue;
211 
212                     Node *lca = sp.LCA(pos[x],pos[y]);
213                     if (pos[x] == lca)  continue;
214 
215                     sp.Cut(pos[x]);
216                     sp.Link(pos[x],pos[y],c);
217 
218                 }
219                 else
220                 {
221                     Node *lca = sp.LCA(pos[x],pos[y]);
222 
223                     if (lca == nil || x == y)
224                         continue;
225 
```

```
                    lca->c[1]->lazy = c;
                    sp.Pushup(lca->c[1]);
                    sp.Pushup(lca);
                    if (pos[x] != lca)
                    {
                        sp.Splay(pos[x]);
                        pos[x]->lazy = c;
                        sp.Pushup(pos[x]);
                    }
                }
            }
        }
    }
    return 0;
}
```

5.3 可持久化线段树

区间第k小数，内存压缩版，POJ2014。

```
#include <cstdio>
#include <algorithm>
using namespace std;

const int MAXN=100000,MAXM=100000;

struct node
{
    node *l,*r;
    int sum;
}tree[MAXN*4+MAXM*20];

int N;
node *newnode()
{
    tree[N].l=tree[N].r=NULL;
    tree[N].sum=0;
    return &tree[N++];
}
node *newnode(node *x)
{
    tree[N].l=x->l;
    tree[N].r=x->r;
    tree[N].sum=x->sum;
    return &tree[N++];
}
node *build(int l,int r)
{
    node *x=newnode();
```

```
30      if (l<r)
31      {
32          int mid=l+r>>1;
33          x->l=build(l,mid);
34          x->r=build(mid+1,r);
35          x->sum=x->l->sum+x->r->sum;
36      }
37      else
38          x->sum=0;
39      return x;
40  }
41  node *update(node *x,int l,int r,int p,int v)
42  {
43      if (l<r)
44      {
45          int mid=l+r>>1;
46          node *nx=newnode(x);
47          if (p<=mid)
48          {
49              node *ret=update(x->l,l,mid,p,v);
50              nx->l=ret;
51          }
52          else
53          {
54              node *ret=update(x->r,mid+1,r,p,v);
55              nx->r=ret;
56          }
57          nx->sum=nx->l->sum+nx->r->sum;
58          return nx;
59      }
60      else
61      {
62          node *nx=newnode(x);
63          nx->sum+=v;
64          return nx;
65      }
66  }
67  int query(node *x1,node *x2,int l,int r,int k)
68  {
69      if (l<r)
70      {
71          int mid=l+r>>1;
72          int lsum=x2->l->sum-x1->l->sum;
73          if (lsum>=k)
74              return query(x1->l,x2->l,l,mid,k);
75          else
76              return query(x1->r,x2->r,mid+1,r,k-lsum);
77      }
```

```
78      else
79          return l;
80  }
81  char s[10];
82  node *root[MAXM+1];
83  int a[MAXN],b[MAXN];
84  int init(int n)
85  {
86      for (int i=0;i<n;i++)
87          b[i]=a[i];
88      sort(b,b+n);
89      int tn=unique(b,b+n)-b;
90      for (int i=0;i<n;i++)
91      {
92          int l=0,r=tn-1;
93          while (l<r)
94          {
95              int mid=l+r>>1;
96              if (b[mid]>=a[i])
97                  r=mid;
98              else
99                  l=mid+1;
100         }
101         a[i]=l;
102     }
103     return tn;
104 }
105 int main()
106 {
107     int cas=1,n;
108     while (scanf("%d",&n)!=EOF)
109     {
110         printf("Case␣%d:\n",cas++);
111         for (int i=0;i<n;i++)
112             scanf("%d",&a[i]);
113         int tn=init(n);
114         N=0;
115         root[0]=build(0,tn-1);
116         for (int i=1;i<=n;i++)
117             root[i]=update(root[i-1],0,tn-1,a[i-1],1);
118         int m;
119         scanf("%d",&m);
120         for (int i=0;i<m;i++)
121         {
122             int s,t;
123             scanf("%d%d",&s,&t);
124             printf("%d\n",b[query(root[s-1],root[t],0,tn-1,t-s
                +2>>1)]);
```

```
125         }
126     }
127     return 0;
128 }
```

5.4 treap正式版

支持翻转。

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <algorithm>
4  using namespace std;
5
6  const int MAXN = 100000;
7  const int MAXM = 100000;
8  const int inf = 0x7fffffff;
9  int a[MAXN];
10 struct Treap
11 {
12     int N;
13     Treap()
14     {
15         N = 0;
16         root = NULL;
17     }
18     void init()
19     {
20         N = 0;
21         root = NULL;
22     }
23     struct Treap_Node
24     {
25         Treap_Node *son[2];//left & right
26         int value, fix;
27         bool lazy;
28         int size;
29         Treap_Node() {}
30         Treap_Node(int _value)
31         {
32             son[0] = son[1] = NULL;
33             value = _value;
34             fix = rand() * rand();
35             lazy = 0;
36             size = 1;
37         }
38         int sonSize(bool flag)
39         {
40             if (son[flag] == NULL)
```

```
41                  return 0;
42              else
43                  return son[flag]->size;
44          }
45      } node[MAXN], *root, *pos[MAXN];
46      void up(Treap_Node *p)
47      {
48          p->size = p->sonSize(0) + p->sonSize(1) + 1;
49      }
50      void down(Treap_Node *p)
51      {
52          if (!p->lazy)
53              return ;
54          for (int i = 0; i < 2; i++)
55              if (p->son[i])
56                  p->son[i]->lazy = !p->son[i]->lazy;
57          swap(p->son[0], p->son[1]);
58          p->lazy = 0;
59      }
60      Treap_Node *merge(Treap_Node *p, Treap_Node *q)
61      {
62          if (p == NULL)
63              return q;
64          else if (q == NULL)
65              return p;
66          if (p->fix <= q->fix)
67          {
68              down(p);
69              p->son[1] = merge(p->son[1], q);
70              up(p);
71              return p;
72          }
73          else
74          {
75              down(q);
76              q->son[0] = merge(p, q->son[0]);
77              up(q);
78              return q;
79          }
80      }
81      pair<Treap_Node *, Treap_Node *> split(Treap_Node *p, int n)
82      {
83          if (p == NULL)
84              return make_pair((Treap_Node *)NULL, (Treap_Node *)
                    NULL);
85          if (!n)
86              return make_pair((Treap_Node *)NULL, p);
87          if (n == p->size)
```

```
88              return make_pair(p, (Treap_Node *)NULL);
89          down(p);
90          if (p->sonSize(0) >= n)
91          {
92              pair<Treap_Node *, Treap_Node *> ret = split(p->son
                    [0], n);
93              p->son[0] = ret.second;
94              up(p);
95              return make_pair(ret.first, p);
96          }
97          else
98          {
99              pair<Treap_Node *, Treap_Node *> ret = split(p->son
                    [1], n - p->sonSize(0) - 1);
100             p->son[1] = ret.first;
101             up(p);
102             return make_pair(p, ret.second);
103         }
104     }
105     int smalls(Treap_Node *p,int value)
106     {
107         if (p==NULL)
108             return 0;
109         if (p->value<=value)
110             return 1+p->sonSize(0)+smalls(p->son[1],value);
111         else
112             return smalls(p->son[0],value);
113     }
114     void insert(int value)
115     {
116         Treap_Node *p = &node[N++];
117         *p = Treap_Node(value);
118         pair<Treap_Node *, Treap_Node *> ret = split(root, smalls
                (root, value));
119         root = merge(merge(ret.first, p), ret.second);
120     }
121     void remove(int value)
122     {
123         pair<Treap_Node *, Treap_Node *> ret = split(root, smalls
                (root, value) - 1);
124         root = merge(ret.first, split(ret.second, 1).second);
125     }
126     Treap_Node *build(int s, int t)
127     {
128         int idx = t + s >> 1;
129         Treap_Node *p = &node[N++];
130         *p = Treap_Node(a[idx]);
131         pos[a[idx]] = p;
```

```
132         if (idx > s)
133             p = merge(build(s, idx - 1), p);
134         if (idx < t)
135             p = merge(p, build(idx + 1, t));
136         up(p);
137         return p;
138     }
139     void build(int n)
140     {
141         root = build(0, n - 1);
142     }
143     void *reverse(int s, int t)
144     {
145         pair<Treap_Node *, Treap_Node *> tmp1, tmp2;
146         tmp1 = split(root, s - 1);
147         tmp2 = split(tmp1.second, t - s + 1);
148         tmp2.first->lazy = !tmp2.first->lazy;
149         root = merge(tmp1.first, merge(tmp2.first, tmp2.second));
150     }
151 };
152 Treap treap;
153 int main()
154 {
155     treap.init();
156     int n;
157     scanf("%d", &n);
158     for (int i = 0; i < n; i++)
159         scanf("%d", &a[i]);
160     treap.build(n);
161 }
```

5.5 树链剖分

5.5.1 点权

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cstdlib>
4  #include <algorithm>
5  using namespace std;
6  const int MAX = 12000;
7  const int LOG = 15;
8  const int oo = 0x3f3f3f3f;
9  struct Edge
10 {
11         int to, w, id;
12         Edge* next;
```

```
13 } memo[MAX << 1], *cur, *g[MAX], *pree[MAX], *solid[MAX], *valid[
       MAX];
14 int dp[MAX][LOG], pos[MAX], lst[MAX], dep[MAX], cnt[MAX], h[MAX],
       K, n;
15 void init()
16 {
17     for (int i = 1; i <= n; i++)
18     {
19         g[i] = NULL;
20         valid[i] = NULL;
21         solid[i] = NULL;
22         pree[i] = NULL;
23     }
24     for (int i = 0; i < LOG; i++)
25     {
26         dp[1][i] = 1;
27     }
28     cur = memo;
29     K = 0;
30 }
31 void add(int u, int v, int w, int id)
32 {
33     cur->to = v;
34     cur->w = w;
35     cur->id = id;
36     cur->next = g[u];
37     g[u] = cur++;
38 }
39 void dfsLCA(int d, int u, int f)
40 {
41     dep[u] = d;
42     dp[u][0] = f;
43     cnt[u] = 1;
44     for (int i = 1; i < LOG; i++)
45     {
46         dp[u][i] = dp[dp[u][i - 1]][i - 1];
47     }
48     for (Edge* it = g[u]; it; it = it->next)
49     {
50         int v = it->to;
51         if (v != f)
52         {
53             pree[v] = it;
54             valid[it->id] = it;
55             dfsLCA(d + 1, v, u); //RE
56             cnt[u] += cnt[v];
57             if (solid[u] == NULL || cnt[solid[u]->to] < cnt[v])
58             {
```

```
59                  solid[u] = it;
60              }
61          }
62      }
63  }
64  void dfsChain(int u, int head)
65  {
66      h[u] = head;
67      if (solid[u])
68      {
69          lst[pos[u] = K++] = u;
70          dfsChain(solid[u]->to, head);
71      }
72      else
73      for (Edge* it = g[u]; it; it = it->next)
74      {
75          int v = it->to;
76          if (it != solid[u] && v != dp[u][0])
77          {
78              dfsChain(v, v);
79          }
80      }
81  }
82  int getLCA(int u, int v)
83  {
84      if (dep[u] < dep[v])
85          swap(u, v);
86      for (int st = 1 << (LOG - 1), i = LOG - 1; i >= 0; i--, st
           >>= 1)
87      {
88          if (st <= dep[u] - dep[v])
89          {
90              u = dp[u][i];
91          }
92      }
93      if (u == v)
94          return u;
95      for (int i = LOG - 1; i >= 0; i--)
96      {
97          if (dp[u][i] != dp[v][i])
98          {
99              u = dp[u][i];
100             v = dp[v][i];
101         }
102     }
103     return dp[u][0];
104 }
105 struct Node
```

```
{
        int l, r, ma, mi;
        bool rev;
} seg[MAX << 2];
void reverse(int k)
{
    seg[k].mi *= -1;
    seg[k].ma *= -1;
    seg[k].rev ^= 1;
    swap(seg[k].mi, seg[k].ma);
}
void pushdown(int k)
{
    if (seg[k].rev)
    {
        reverse(k << 1);
        reverse(k << 1 | 1);
        seg[k].rev = false;
    }
}
void update(int k)
{
    seg[k].mi = min(seg[k << 1].mi, seg[k << 1 | 1].mi);
    seg[k].ma = max(seg[k << 1].ma, seg[k << 1 | 1].ma);
}
void init(int k, int l, int r)
{
    seg[k].l = l;
    seg[k].r = r;
    seg[k].rev = false;
    if (l == r)
    {
        seg[k].mi = seg[k].ma = solid[lst[l]]->w; //solid WA
        return;
    }
    int mid = l + r >> 1;
    init(k << 1, l, mid);
    init(k << 1 | 1, mid + 1, r);
    update(k);
}
void update(int k, int id, int v)
{
    if (seg[k].l == seg[k].r)
    {
        seg[k].mi = seg[k].ma = solid[lst[id]]->w = v;
        return;
    }
    pushdown(k);
```

```
154     int mid = seg[k].l + seg[k].r >> 1;
155     if (id <= mid)
156         update(k << 1, id, v);
157     else
158         update(k << 1 | 1, id, v);
159     update(k);
160 }
161 void reverse(int k, int l, int r)
162 {
163     if (seg[k].l > r || seg[k].r < l)
164         return;
165     if (seg[k].l >= l && seg[k].r <= r)
166     {
167         reverse(k);
168         return;
169     }
170     pushdown(k);
171     reverse(k << 1, l, r);
172     reverse(k << 1 | 1, l, r);
173     update(k);
174 }
175 int read(int k, int l, int r)
176 {
177     if (seg[k].l > r || seg[k].r < l)
178         return -oo;
179     if (seg[k].l >= l && seg[k].r <= r)
180         return seg[k].ma;
181     pushdown(k);
182     return max(read(k << 1, l, r), read(k << 1 | 1, l, r));
183 }
184 void setEdge(int id, int v)
185 {
186     Edge* it = valid[id];
187     if (h[it->to] != it->to)
188     {
189         update(1, pos[dp[it->to][0]], v);
190     }
191     else
192     {
193         it->w = v;
194     }
195 }
196 void negateLCA(int t, int u)
197 {
198     while (t != u)
199     {
200         int tmp = h[u];
201         if (dep[tmp] < dep[t])
```

```
202             tmp = t;
203         if (h[u] == u)
204         {
205             pree[u]->w *= -1;
206             u = dp[u][0];
207         }
208         else
209         {
210             reverse(1, pos[tmp], pos[dp[u][0]]);
211             u = tmp;
212         }
213     }
214 }
215 void negate(int u, int v)
216 {
217     int t = getLCA(u, v);
218     negateLCA(t, u);
219     negateLCA(t, v);
220 }
221 int maxLCA(int t, int u)
222 {
223     int ret = -oo;
224     while (t != u)
225     {
226         int tmp = h[u];
227         if (dep[tmp] < dep[t])
228             tmp = t;
229         if (h[u] == u)
230         {
231             ret = max(ret, pree[u]->w);
232             u = dp[u][0];
233         }
234         else
235         {
236             ret = max(ret, read(1, pos[tmp], pos[dp[u][0]]));
237             u = tmp;
238         }
239     }
240     return ret;
241 }
242 int query(int u, int v)
243 {
244     int t = getLCA(u, v);
245     return max(maxLCA(t, u), maxLCA(t, v));
246 }
247 int main()
248 {
249     int T;
```

```
250     int u, v, w;
251     char op[15];
252     scanf("%d", &T);
253     while (T--)
254     {
255         scanf("%d", &n);
256         init();
257         for (int i = 1; i < n; i++)
258         {
259             scanf("%d%d%d", &u, &v, &w);
260             add(u, v, w, i);
261             add(v, u, w, i);
262         }
263         dfsLCA(0, 1, 1);
264         dfsChain(1, 1);
265         init(1, 0, K - 1);
266         while (scanf("%s", op), op[0] != 'D')
267         {
268             scanf("%d%d", &u, &v);
269             if (op[0] == 'C')
270             {
271                 setEdge(u, v);
272             }
273             else if (op[0] == 'N')
274             {
275                 negate(u, v);
276             }
277             else
278             {
279                 printf("%d\n", query(u, v));
280             }
281         }
282     }
283     return 0;
284 }
```

5.5.2 边权

```
1  #include <cstdio>
2  #include <iostream>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <cmath>
6  #include <cstring>
7  using namespace std;
8  int n,m,sum,pos;
9  int head[50005],e;
10 int s[50005],from[50005];
11 int fa[50005][20],deep[50005],num[50005];
```

```
12 int solid[50005],p[50005],fp[50005];
13 struct N
14 {
15   int l,r,mid;
16   int add,w;
17 }nod[50005*4];
18 struct M
19 {
20   int v,next;
21 }edge[100005];
22 void addedge(int u,int v)
23 {
24   edge[e].v=v;
25   edge[e].next=head[u];
26   head[u]=e++;
27
28   edge[e].v=u;
29   edge[e].next=head[v];
30   head[v]=e++;
31 }
32 void LCA(int st,int f,int d)
33 {
34   deep[st]=d;
35   fa[st][0]=f;
36   num[st]=1;
37   int i,v;
38   for(i=1;i<20;i++)
39     fa[st][i]=fa[fa[st][i-1]][i-1];
40   for(i=head[st];i!=-1;i=edge[i].next)
41   {
42     v=edge[i].v;
43     if(v!=f)
44     {
45       LCA(v,st,d+1);
46       num[st]+=num[v];
47       if(solid[st]==-1||num[v]>num[solid[st]])
48         solid[st]=v;
49     }
50   }
51 }
52 void getpos(int st,int sp)
53 {
54   from[st]=sp;
55   if(solid[st]!=-1)
56   {
57     p[st]=pos++;
58     fp[p[st]]=st;
59     getpos(solid[st],sp);
```

```
60    }
61    else
62    {
63      p[st]=pos++;
64      fp[p[st]]=st;
65      return;
66    }
67    int i,v;
68    for(i=head[st];i!=-1;i=edge[i].next)
69    {
70      v=edge[i].v;
71      if(v!=solid[st]&&v!=fa[st][0])
72        getpos(v,v);
73    }
74  }
75  int getLCA(int u,int v)
76  {
77    if(deep[u]<deep[v])
78      swap(u,v);
79    int d=1<<19,i;
80    for(i=19;i>=0;i--)
81    {
82      if(d<=deep[u]-deep[v])
83        u=fa[u][i];
84      d>>=1;
85    }
86    if(u==v)
87      return u;
88    for(i=19;i>=0;i--)
89      if(fa[u][i]!=fa[v][i])
90      {
91        u=fa[u][i];
92        v=fa[v][i];
93      }
94    return fa[u][0];
95  }
96  void init(int p,int l,int r)
97  {
98    nod[p].l=l;
99    nod[p].r=r;
100   nod[p].mid=(l+r)>>1;
101   nod[p].add=0;
102   if(l==r)
103     nod[p].w=s[fp[l]];
104   else
105   {
106     init(p<<1,l,nod[p].mid);
107     init(p<<1|1,nod[p].mid+1,r);
```

```
108   }
109 }
110 void lazy(int p)
111 {
112   if(nod[p].add!=0)
113   {
114     nod[p<<1].add+=nod[p].add;
115     nod[p<<1|1].add+=nod[p].add;
116     nod[p].add=0;
117   }
118 }
119 void update(int p,int l,int r,int v)
120 {
121   if(nod[p].l==l&&nod[p].r==r)
122   {
123     nod[p].add+=v;
124     return;
125   }
126   lazy(p);
127   if(nod[p].mid<l)
128     update(p<<1|1,l,r,v);
129   else if(nod[p].mid>=r)
130     update(p<<1,l,r,v);
131   else
132   {
133     update(p<<1,l,nod[p].mid,v);
134     update(p<<1|1,nod[p].mid+1,r,v);
135   }
136 }
137 int read(int p,int l,int r)
138 {
139   if(nod[p].l==l&&nod[p].r==r)
140     return nod[p].w+nod[p].add;
141   lazy(p);
142   if(nod[p].mid<l)
143     return read(p<<1|1,l,r);
144   else if(nod[p].mid>=r)
145     return read(p<<1,l,r);
146 }
147 void jump(int st,int ed,int val)
148 {
149   while(deep[st]>=deep[ed])
150   {
151     int tmp=from[st];
152     if(deep[tmp]<deep[ed])
153       tmp=ed;
154     update(1,p[tmp],p[st],val);
155     st=fa[tmp][0];
```

```
156   }
157 }
158 void change(int st,int ed,int val)
159 {
160   int lca=getLCA(st,ed);
161   jump(st,lca,val);
162   jump(ed,lca,val);
163   jump(lca,lca,-val);
164 }
165 int main()
166 {
167   while(scanf("%d%d%d",&n,&m,&sum)==3)
168   {
169     int i;
170     s[0]=0;pos=0;deep[0]=-1;
171     memset(fa,0,sizeof(fa));
172     for(i=1;i<=n;i++)
173     {
174       solid[i]=-1;
175       scanf("%d",&s[i]);
176     }
177     memset(head,-1,sizeof(head));
178     e=0;
179     for(i=0;i<m;i++)
180     {
181       int a,b;
182       scanf("%d%d",&a,&b);
183       addedge(a,b);
184     }
185     LCA(1,0,0);
186     getpos(1,1);
187     init(1,0,pos-1);
188     for(i=0;i<sum;i++)
189     {
190       char que[5];
191       scanf("%s",que);
192       if(que[0]!='Q')
193       {
194         int a,b,c;
195         scanf("%d%d%d",&a,&b,&c);
196         if(que[0]=='D')
197           c=-c;
198         change(a,b,c);
199       }
200       else
201       {
202         int a;
203         scanf("%d",&a);
```

```
204         printf("%d\n",read(1,p[a],p[a]));
205       }
206     }
207   }
208   return 0;
209 }
```

5.6 划分树

```
1  int n,m;
2  struct elem
3  {
4      int v,index;
5  }a[120000];
6  int d[30][120000];
7  int s[30][120000];
8
9  bool cmp(elem a,elem b)
10 {
11     if (a.v == b.v)
12         return a.index <= b.index;
13     return a.v < b.v;
14 }
15
16 void build(int depth,int l,int r)
17 {
18     if (l == r)
19         return;
20     int mid = (l+r)/2;
21     int tl,tr;
22     tl = tr = 0;
23     for (int i = l;i <= r;i++)
24     {
25         if (cmp(a[d[depth][i]],a[mid]))
26         {
27             d[depth+1][l+tl] = d[depth][i];
28             tl++;
29         }
30         else
31         {
32             d[depth+1][mid+1+tr] = d[depth][i];
33             tr++;
34         }
35         s[depth][i] = tl;
36     }
37     build(depth+1,l,mid);
38     build(depth+1,mid+1,r);
39 }
40
```

```
41 int find(int depth,int dl,int dr,int fl,int fr,int k)
42 {
43     if (fl == fr)
44         return a[d[depth][fl]].v;
45     int ls,rs;
46     int mid = (dl+dr)/2;
47     ls = (fl == dl)? 0 : s[depth][fl-1];
48     rs = s[depth][fr];
49     return (rs-ls < k)? find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+
          fr-dl-rs+1,k-(rs-ls)) : find(depth+1,dl,mid,dl+ls,dl+rs-1,
          k);
50 }
51
52 int main()
53 {
54     while (scanf("%d%d",&n,&m) != EOF)
55     {
56         for (int i = 1;i <= n;i++)
57         {
58             scanf("%d",&a[i].v);
59             a[i].index = i;
60         }
61         sort(a+1,a+n+1,cmp);
62         for (int i = 1;i <= n;i++)
63             d[0][a[i].index] = i;
64         build(0,1,n);
65         int l,r,k;
66         for (int i = 1;i <= m;i++)
67         {
68             scanf("%d%d%d",&l,&r,&k);
69             printf("%d\n",find(0,1,n,l,r,k));
70         }
71     }
72     return 0;
73 }
```

5.7 树状数组

```
1 int read(int k)
2 {
3     int sum = 0;
4     for (; k; k^=k&-k)
5         sum+=tree[k];
6     return sum;
7 }
8 void update(int k, int v)
9 {
10    for (; k<=MaxN; k+=k&-k)
11        tree[k]+=v;
```

```
12 }
13 int find_Kth(int k)
14 {
15     int idx = 0;
16     for(int i=20; i>=0; i--)
17     {
18         idx |= 1 << i;
19         if(idx <= MaxN && tree[idx] < k)
20             k -= tree[idx];
21         else  idx ^= 1 << i;
22     }
23     return idx + 1;
24 }
```

6 图论

6.1 优先队列优化的dijkstra

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int MAXN=100;
const int MAXM=1000;
int N,L;
int head[MAXN];
struct edges
{
    int to,next,cost;
} edge[MAXM];
int dist[MAXN];
class states
{
public:
    int cost,id;
};
class cmp
{
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost>j.cost;
    }
};
void init(int n)
{
    N=n;
    L=0;
    for (int i=0; i<n; i++)
        head[i]=-1;
}
void add_edge(int x,int y,int cost)
{
    edge[L].to=y;
    edge[L].cost=cost;
    edge[L].next=head[x];
    head[x]=L++;
}
int dijkstra(int s,int t)
```

```
{
    memset(dist,63,sizeof(dist));
    states u;
    u.id=s;
    u.cost=0;
    dist[s]=0;
    priority_queue<states,vector<states>,cmp> q;
    q.push(u);
    while (!q.empty())
    {
        u=q.top();
        q.pop();
        if (u.id==t) return dist[t];
        if (u.cost!=dist[u.id]) continue;
        for (int i=head[u.id]; i!=-1; i=edge[i].next)
        {
            states v=u;
            v.id=edge[i].to;
            if (dist[v.id]>dist[u.id]+edge[i].cost)
            {
                v.cost=dist[v.id]=dist[u.id]+edge[i].cost;
                q.push(v);
            }
        }
    }
    return -1;
}
int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    init(n);
    for (int i=0; i<m; i++)
    {
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        add_edge(x,y,z);
        add_edge(y,x,z);
    }
    int s,t;
    scanf("%d%d",&s,&t);
    printf("%d\n",dijkstra(s,t));
    return 0;
}
```

6.2 SAP四版

```
const int MAXEDGE=20400;
const int MAXN=400;
```

```
3  const int inf=0x3fffffff;
4  struct edges
5  {
6      int cap,to,next,flow;
7  } edge[MAXEDGE+100];
8  struct nodes
9  {
10     int head,label,pre,cur;
11 } node[MAXN+100];
12 int L,N;
13 int gap[MAXN+100];
14 void init(int n)
15 {
16     L=0;
17     N=n;
18     for (int i=0; i<N; i++)
19         node[i].head=-1;
20 }
21 void add_edge(int x,int y,int z,int w)
22 {
23     edge[L].cap=z;
24     edge[L].flow=0;
25     edge[L].to=y;
26     edge[L].next=node[x].head;
27     node[x].head=L++;
28     edge[L].cap=w;
29     edge[L].flow=0;
30     edge[L].to=x;
31     edge[L].next=node[y].head;
32     node[y].head=L++;
33 }
34 int maxflow(int s,int t)
35 {
36     memset(gap,0,sizeof(gap));
37     gap[0]=N;
38     int u,ans=0;
39     for (int i=0; i<N; i++)
40     {
41         node[i].cur=node[i].head;
42         node[i].label=0;
43     }
44     u=s;
45     node[u].pre=-1;
46     while (node[s].label<N)
47     {
48         if (u==t)
49         {
50             int min=inf;
```

```
51              for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].
                    pre)
52                  if (min>edge[i].cap-edge[i].flow)
53                      min=edge[i].cap-edge[i].flow;
54              for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].
                    pre)
55              {
56                  edge[i].flow+=min;
57                  edge[i^1].flow-=min;
58              }
59              u=s;
60              ans+=min;
61              continue;
62          }
63          bool flag=false;
64          int v;
65          for (int i=node[u].cur; i!=-1; i=edge[i].next)
66          {
67              v=edge[i].to;
68              if (edge[i].cap-edge[i].flow && node[v].label+1==node
                    [u].label)
69              {
70                  flag=true;
71                  node[u].cur=node[v].pre=i;
72                  break;
73              }
74          }
75          if (flag)
76          {
77              u=v;
78              continue;
79          }
80          node[u].cur=node[u].head;
81          int min=N;
82          for (int i=node[u].head; i!=-1; i=edge[i].next)
83              if (edge[i].cap-edge[i].flow && node[edge[i].to].
                    label<min)
84                  min=node[edge[i].to].label;
85          gap[node[u].label]--;
86          if (!gap[node[u].label]) return ans;
87          node[u].label=min+1;
88          gap[node[u].label]++;
89          if (u!=s) u=edge[node[u].pre^1].to;
90      }
91      return ans;
92  }
```

6.3 费用流三版

T了可以改成栈。

```
const int MAXM=60000;
const int MAXN=400;
const int inf=0x3fffffff;
int L,N;
int K;
struct edges
{
    int to,next,cap,flow,cost;
} edge[MAXM];
struct nodes
{
    int dis,pre,head;
    bool visit;
} node[MAXN];
void init(int n)
{
    N=n;
    L=0;
    for (int i=0; i<N; i++)
        node[i].head=-1;
}
void add_edge(int x,int y,int cap,int cost)
{
    edge[L].to=y;
    edge[L].cap=cap;
    edge[L].cost=cost;
    edge[L].flow=0;
    edge[L].next=node[x].head;
    node[x].head=L++;
    edge[L].to=x;
    edge[L].cap=0;
    edge[L].cost=-cost;
    edge[L].flow=0;
    edge[L].next=node[y].head;
    node[y].head=L++;
}
bool spfa(int s,int t)
{
    queue <int> q;
    for (int i=0; i<N; i++)
    {
        node[i].dis=0x3fffffff;
        node[i].pre=-1;
        node[i].visit=0;
```

```
45      }
46      node[s].dis=0;
47      node[s].visit=1;
48      q.push(s);
49      while (!q.empty())
50      {
51          int u=q.front();
52          node[u].visit=0;
53          for (int i=node[u].head; i!=-1; i=edge[i].next)
54          {
55              int v=edge[i].to;
56              if (edge[i].cap>edge[i].flow &&
57                      node[v].dis>node[u].dis+edge[i].cost)
58              {
59                  node[v].dis=node[u].dis+edge[i].cost;
60                  node[v].pre=i;
61                  if (!node[v].visit)
62                  {
63                      node[v].visit=1;
64                      q.push(v);
65                  }
66              }
67          }
68          q.pop();
69      }
70      if (node[t].pre==-1)
71          return 0;
72      else
73          return 1;
74  }
75  int mcmf(int s,int t,int &cost)
76  {
77      int flow=0;
78      while (spfa(s,t))
79      {
80          int max=inf;
81          for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
82          {
83              if (max>edge[i].cap-edge[i].flow)
84                  max=edge[i].cap-edge[i].flow;
85          }
86          for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
87          {
88              edge[i].flow+=max;
89              edge[i^1].flow-=max;
90              cost+=edge[i].cost*max;
91          }
92          flow+=max;
```

```
93      }
94      return flow;
95  }
```

6.4 匈牙利

6.4.1 新版,隐式图可解

```
1   bool check(int u)
2   {
3       for (int i=head[u]; i!=-1; i=edge[i].next)
4       {
5           int v=edge[i].to;
6           if (matc[v]==u) continue;
7           if (!use[v])
8           {
9               use[v]=1;
10              if (matc[v]==-1 || check(matc[v]))
11              {
12                  matc[v]=u;
13                  matc[u]=v;
14                  return 1;
15              }
16          }
17      }
18      return 0;
19  }
20  int match()
21  {
22      int ret=0;
23      memset(matc,-1,sizeof(matc));
24      for (int u=0; u<N; u++)
25      {
26          if (matc[u]!=-1) continue;
27          memset(use,0,sizeof(use));
28          if (check(u))
29              ret++;
30      }
31      return ret;
32  }
```

6.4.2 邻接矩阵

```
1   bool check(int u)
2   {
3       for (int v=0; v<N; v++)
4           if (am[u][v] && !use[v])
5           {
6               use[v]=1;
```

```
7               if (pre[v]==-1 || check(pre[v]))
8               {
9                   pre[v]=u;
10                  return 1;
11              }
12          }
13      return 0;
14  }
15  int match()
16  {
17      int ret=0;
18      memset(pre,-1,sizeof(pre));
19      for (int u=0; u<N; u++)
20      {
21          memset(use,0,sizeof(use));
22          if (check(u))
23              ret++;
24      }
25      return ret;
26  }
```

6.4.3 邻接表

```
1   bool check(int u)
2   {
3       for (int i=head[u]; i!=-1; i=edge[i].next)
4       {
5           int v=edge[i].to;
6           if (!use[v])
7           {
8               use[v]=1;
9               if (pre[v]==-1 || check(pre[v]))
10              {
11                  pre[v]=u;
12                  return 1;
13              }
14          }
15      }
16      return 0;
17  }
18  int match()
19  {
20      int ret=0;
21      memset(pre,-1,sizeof(pre));
22      for (int u=1; u<=N; u++)
23      {
24          memset(use,0,sizeof(use));
25          if (check(u))
26              ret++;
```

```
27      }
28      return ret;
29  }
```

6.5 一般图匹配带花树

```
1   const int MaxN = 222;
2   int N;
3   bool Graph[MaxN+1][MaxN+1];
4   int Match[MaxN+1];
5   bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
6   int Head,Tail;
7   int Queue[MaxN+1];
8   int Start,Finish;
9   int NewBase;
10  int Father[MaxN+1],Base[MaxN+1];
11  int Count;
12  void CreateGraph()
13  {
14      int u,v;
15      memset(Graph,false,sizeof(Graph));
16      scanf("%d",&N);
17      while (scanf("%d%d",&u,&v) != EOF)
18          Graph[u][v] = Graph[v][u] = true;
19  }
20  void Push(int u)
21  {
22      Queue[Tail] = u;
23      Tail++;
24      InQueue[u] = true;
25  }
26  int Pop()
27  {
28      int res = Queue[Head];
29      Head++;
30      return res;
31  }
32  int FindCommonAncestor(int u,int v)
33  {
34      memset(InPath,false,sizeof(InPath));
35      while (true)
36      {
37          u = Base[u];
38          InPath[u] = true;
39          if (u == Start) break;
40          u = Father[Match[u]];
41      }
42      while (true)
43      {
```

```
44          v = Base[v];
45          if (InPath[v]) break;
46          v = Father[Match[v]];
47      }
48      return v;
49  }
50  void ResetTrace(int u)
51  {
52      int v;
53      while (Base[u] != NewBase)
54      {
55          v = Match[u];
56          InBlossom[Base[u]] = InBlossom[Base[v]] = true;
57          u = Father[v];
58          if (Base[u] != NewBase) Father[u] = v;
59      }
60  }
61  void BlossomContract(int u,int v)
62  {
63      NewBase = FindCommonAncestor(u,v);
64      memset(InBlossom,false,sizeof(InBlossom));
65      ResetTrace(u);
66      ResetTrace(v);
67      if (Base[u] != NewBase) Father[u] = v;
68      if (Base[v] != NewBase) Father[v] = u;
69      for (int tu = 1; tu <= N; tu++)
70          if (InBlossom[Base[tu]])
71          {
72              Base[tu] = NewBase;
73              if (!InQueue[tu]) Push(tu);
74          }
75  }
76  void FindAugmentingPath()
77  {
78      memset(InQueue,false,sizeof(InQueue));
79      memset(Father,0,sizeof(Father));
80      for (int i = 1; i <= N; i++)
81          Base[i] = i;
82      Head = Tail = 1;
83      Push(Start);
84      Finish = 0;
85      while (Head < Tail)
86      {
87          int u = Pop();
88          for (int v = 1; v <= N; v++)
89              if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u]
                    != v))
90              {
```

```
91                  if ((v == Start) || ((Match[v] > 0) && (Father[
                       Match[v]] > 0)))
92                      BlossomContract(u,v);
93                  else if (Father[v] == 0)
94                  {
95                      Father[v] = u;
96                      if (Match[v] > 0)
97                          Push(Match[v]);
98                      else
99                      {
100                         Finish = v;
101                         return;
102                     }
103                 }
104             }
105     }
106 }
107 void AugmentPath()
108 {
109     int u,v,w;
110     u = Finish;
111     while (u > 0)
112     {
113         v = Father[u];
114         w = Match[v];
115         Match[v] = u;
116         Match[u] = v;
117         u = w;
118     }
119 }
120 void Edmonds()
121 {
122     memset(Match,0,sizeof(Match));
123     for (int u = 1; u <= N; u++)
124         if (Match[u] == 0)
125         {
126             Start = u;
127             FindAugmentingPath();
128             if (Finish > 0) AugmentPath();
129         }
130 }
131 void PrintMatch()
132 {
133     for (int u = 1; u <= N; u++)
134         if (Match[u] > 0)
135             Count++;
136     printf("%d\n",Count);
137     for (int u = 1; u <= N; u++)
```

```
138         if (u < Match[u])
139             printf("%d␣%d\n",u,Match[u]);
140 }
141 int main()
142 {
143     CreateGraph();
144     Edmonds();
145     PrintMatch();
146 }
```

6.6 KM

6.6.1 最大加权匹配

```
1  bool visx[N],visy[N];//x,y中的点是否被访问
2  int lx[N],ly[N];//x,y中的点的标号
3  int matchy[N];//y中各点匹配状态
4  int map[N][N];//二分图描述[x][y]
5  bool find(int x)
6  {
7    visx[x]=true;
8    int t;
9    for (int y=0;y<ycnt;y++)
10   {
11     if (!visy[y])
12     {
13       t=lx[x]+ly[y]-map[x][y];
14       if (t==0)
15       {
16         visy[y]=true;
17         if (matchy[y]==-1 || find(matchy[y]))
18         {
19           matchy[y]=x;
20           return true;
21         }
22       }
23       else if (lack>t) lack=t;
24     }
25   }
26   return false;
27 }
28 void KM()
29 {
30   memset(lx,0,sizeof(lx));
31   memset(ly,0,sizeof(ly));
32   memset(matchy,-1,sizeof(matchy));
33   for (int i=0;i<xcnt;i++)
34     for (int j=0;j<ycnt;j++)
35       if (map[i][j]>lx[i])
```

```
36          lx[i]=map[i][j];
37    for (int x=0;x<xcnt;x++)
38    {
39      while (true)
40      {
41        memset(visx,false,sizeof(visx));
42        memset(visy,false,sizeof(visy));
43        lack=INFI;
44        if (find(x)) break;
45        for (int i=0;i<xcnt;i++)
46        {
47          if (visx[i]) lx[i]-=lack;
48          if (visy[i]) ly[i]+=lack;
49        }
50      }
51    }
52    int cost=0;
53    for (int i=0;i<ycnt;i++)
54      cost+=map[matchy[i]][i];
55  }
```

6.6.2 自认为正确的Kuhn_Munkras

未验证

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  using namespace std;
5  const int MAXN=100;
6  const int inf=0x3f3f3f3f;
7  bool visitx[MAXN],visity[MAXN];
8  int labx[MAXN],laby[MAXN],matx[MAXN],maty[MAXN],slack[MAXN];
9  int ma[MAXN][MAXN];
10 bool check(int x,int n)
11 {
12     visitx[x]=1;
13     for (int i=0; i<n; i++)
14         if (!visity[i])
15             if (labx[x]+laby[i]==ma[x][i])
16             {
17                 visity[i]=1;
18                 if (maty[i]==-1 || check(maty[i],n))
19                 {
20                     matx[x]=i;
21                     maty[i]=x;
22                     return 1;
23                 }
24             }
25             else
```

```
26                  slack[i]=min(slack[i],labx[x]+laby[i]-ma[x][i]);
27
28      return 0;
29  }
30  void maintain(int n)
31  {
32      int diff=inf;
33      for (int i=0; i<n; i++)
34          if (!visity[i])
35              diff=min(diff,slack[i]);
36      for (int i=0; i<n; i++)
37      {
38          if (visitx[i])
39              labx[i]-=diff;
40          if (visity[i])
41              laby[i]+=diff;
42          else
43              slack[i]-=diff;
44      }
45  }
46  int Kuhn_Munkras(int n)
47  {
48      for (int i=0; i<n; i++)
49      {
50          labx[i]=-inf;
51          for (int j=0; j<n; j++)
52              labx[i]=max(labx[i],ma[i][j]);
53      }
54      memset(laby,0,4*n);
55      memset(matx,-1,4*n);
56      memset(maty,-1,4*n);
57      for (int i=0; i<n; i++)
58      {
59          memset(visitx,0,n);
60          memset(visity,0,n);
61          memset(slack,63,4*n);
62          while (!check(i,n))
63          {
64              maintain(n);
65              memset(visitx,0,n);
66              memset(visity,0,n);
67          }
68      }
69      int ret=0;
70      for (int i=0;i<n;i++)
71          ret+=labx[i]+laby[i];
72      return ret;
73  }
```

```
74 int main()
75 {
76     int n,m;
77     scanf("%d%d",&m,&n);
78     for (int i=m; i<n; i++)
79         for (int j=0; j<n; j++)
80             ma[i][j]=0;
81     for (int i=0; i<m; i++)
82         for (int j=0; j<n; j++)
83             scanf("%d",&ma[i][j]);
84     printf("%d\n",Kuhn_Munkras(n));
85     printf("%d",matx[0]+1);
86     for (int i=1;i<m;i++)
87         printf(" %d",matx[i]+1);
88     puts("");
89     return 0;
90 }
```

6.7 *二维平面图的最大流

待整理

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  #include <vector>
6  #include <cmath>
7  #include <map>
8  #include <queue>
9  using namespace std;
10
11 const int maxn = 100100;
12 const int inf = 0x3f3f3f3f;
13 struct Point
14 {
15     int x,y,id;
16     double theta;
17     Point() {}
18     Point(int _x,int _y)
19     {
20         x = _x;
21         y = _y;
22     }
23     Point(Point _s,Point _e,int _id)
24     {
25         id = _id;
26         x = _s.x-_e.x;
```

```
27          y = _s.y-_e.y;
28          theta = atan2(y,x);
29      }
30      bool operator < (const Point &b)const
31      {
32          return theta < b.theta;
33      }
34  };
35
36  map<pair<int,int>,int > idmap;
37  struct Edge
38  {
39      int from,to,next,cap,near,mark;
40  };
41  Edge edge[maxn*2];
42  int head[maxn],L;
43  int cntd[maxn];
44  void addedge(int u,int v,int cap)
45  {
46      cntd[u]++;
47      cntd[v]++;
48      idmap[make_pair(u,v)] = L;
49      edge[L].from = u;
50      edge[L].to = v;
51      edge[L].cap = cap;
52      edge[L].next = head[u];
53      edge[L].mark = -1;
54      head[u] = L++;
55  }
56
57  int rtp[maxn];
58  Point p[maxn],tp[maxn];
59  int n,m,S,T;
60  int vid;
61
62  struct Edge2
63  {
64      int to,next,dis;
65  } edge2[maxn*2];
66  int head2[maxn],L2;
67
68  void addedge2(int u,int v,int dis)
69  {
70      edge2[L2].to = v;
71      edge2[L2].dis = dis;
72      edge2[L2].next = head2[u];
73      head2[u] = L2++;
74  }
```

```
75
76  int dist[maxn];
77  bool inq[maxn];
78  int SPFA(int s,int t)
79  {
80      queue<int> Q;
81      memset(inq,false,sizeof(inq));
82      memset(dist,63,sizeof(dist));
83      Q.push(s);
84      dist[s] = 0;
85      while (!Q.empty())
86      {
87          int now = Q.front();
88          Q.pop();
89          for (int i = head2[now]; i != -1; i = edge2[i].next)
90              if (dist[edge2[i].to] > dist[now]+edge2[i].dis)
91              {
92                  dist[edge2[i].to] = dist[now]+edge2[i].dis;
93                  if (inq[edge2[i].to] == false)
94                  {
95                      inq[edge2[i].to] = true;
96                      Q.push(edge2[i].to);
97                  }
98              }
99          inq[now] = false;
100     }
101     return dist[t];
102 }
103
104 int main()
105 {
106     int totcas;
107     scanf("%d",&totcas);
108     for (int cas = 1; cas <= totcas; cas++)
109     {
110         idmap.clear();
111         L = 0;
112         scanf("%d%d",&n,&m);
113         S = T = 0;
114         for (int i = 0; i < n; i++)
115         {
116             head[i] = -1;
117             scanf("%d%d",&p[i].x,&p[i].y);
118             if (p[S].x > p[i].x)
119                 S = i;
120             if (p[T].x < p[i].x)
121                 T = i;
122             cntd[i] = 0;
```

```
123         }
124         //源汇中间加入一个特殊节点
125         head[n] = -1;
126         n ++;
127         addedge(S,n-1,inf);
128         addedge(n-1,S,inf);
129         addedge(T,n-1,inf);
130         addedge(n-1,T,inf);
131
132         for (int i = 0; i < m; i++)
133         {
134             int u,v,cap;
135             scanf("%d%d%d",&u,&v,&cap);
136             u--;
137             v--;
138             addedge(u,v,cap);
139             addedge(v,u,cap);
140         }
141
142         for (int i = 0; i < n; i++)
143         {
144             int tot = 0;
145             //源点汇点连到特殊点的方向需要特别考虑一下
146             if (i == S)
147                 tp[tot++] = Point(Point(0,0),Point(-1,0),n-1);
148             else if (i == T)
149                 tp[tot++] = Point(Point(0,0),Point(1,0),n-1);
150             else if (i == n-1)
151             {
152                 tp[tot++] = Point(Point(0,0),Point(1,0),S);
153                 tp[tot++] = Point(Point(0,0),Point(-1,0),T);
154             }
155             if (i < n-1)
156             {
157                 for (int j = head[i]; j != -1; j = edge[j].next)
158                 {
159                     if (i == S && edge[j].to == n-1)  continue;
160                     if (i == T && edge[j].to == n-1)  continue;
161                     tp[tot++] = Point(p[i],p[edge[j].to],edge[j].
                            to);
162                 }
163             }
164             sort(tp,tp+tot);
165             for (int j = 0; j < tot; j++)
166                 rtp[tp[j].id] = j;
167             for (int j = head[i]; j != -1; j = edge[j].next)
168                 edge[j].near = tp[(rtp[edge[j].to]+1)%tot].id;
169         }
```

```
170
171         vid = 0;
172         for (int i = 0;i < L;i++)
173             if (edge[i].mark == -1)
174             {
175                 int now = edge[i].from;
176                 int eid = i;
177                 int to  = edge[i].to;
178                 while (true)
179                 {
180                     edge[eid].mark = vid;
181                     eid ^= 1;
182                     now = to;
183                     to = edge[eid].near;
184                     eid = idmap[make_pair(now,to)];
185
186                     if (now == edge[i].from)    break;
187                 }
188                 vid++;
189             }
190
191         L2 = 0;
192         for (int i = 0; i < vid; i++)
193             head2[i] = -1;
194         for (int i = 0; i < L; i++)
195             addedge2(edge[i].mark,edge[i^1].mark,edge[i].cap);
196         printf("%d\n",SPFA(edge[0].mark,edge[1].mark));
197     }
198     return 0;
199 }
```

6.8 强联通

```
1  int dfsnum[2000];
2  int low[2000];
3  int stack[2000];
4  int top;
5  int ans;
6  int an;
7  int be[2000];
8  int flag[2000];
9  void dfs(int x)
10 {
11     dfsnum[x] = low[x] = ans++;
12     stack[++top] = x;
13     flag[x] = 1;
14     for (int i = head[x];i != -1;i = edge[i].next)
15     {
```

```
16          int y = edge[i].to;
17          if (dfsnum[y] == -1)
18          {
19              dfs(y);
20              low[x] = min(low[x],low[y]);
21          }
22          else if (flag[y] == 1)
23              low[x] = min(low[x],dfsnum[y]);
24      }
25      if (dfsnum[x] == low[x])
26      {
27          while (stack[top] != x)
28          {
29              flag[stack[top]] = 0;
30              be[stack[top]] = an;
31              top--;
32          }
33          flag[x] = 0;
34          be[x] = an++;
35          top--;
36      }
37  }
```

调用:

```
1   void SC()
2   {
3       memset(dfsnum,-1,sizeof(dfsnum));
4       memset(flag,0,sizeof(flag));
5       top = 0;
6       an = 0;
7       ans = 0;
8       for (int i = 0;i < n;i++)
9           if (dfsnum[i] == -1)
10              dfs(i);
11  }
```

6.9 最大团以及相关知识

独立集： 独立集是指图的顶点集的一个子集,该子集的导出子图不含边.如果一个独立集不是任何一个独立集的子集, 那么称这个独立集是一个极大独立集.一个图中包含顶点数目最多的独立集称为最大独立集。最大独立集 一定是极大独立集，但是极大独立集不一定是最大的独立集。

支配集： 与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设S是图G的一个支配集，则对于图中的任意一个顶点u，要么属于集合s, 要么与s中的顶点相邻。在s中除去任何元素后s不再是支配集，则支配集s是极小支配集。称G的所有支配集中顶点个数最 少的支配集为最小支配集，最小支配集中的顶点个数成为支配数。

最小点的覆盖： 最小点的覆盖也是图的顶点集的一个子集，如果我们选中一个点，则称这个点将以他为端点的所有边都覆盖了。将图中所有的边都覆盖所用顶点数最少，这个集合就是最小的点的覆盖。

最大团： 图G的顶点的子集，设D是最大团，则D中任意两点相邻。若u，v是最大团，则u,v有边相连，其补图u,v没有边相连，所以图G的最大团=其补图的最大独立集。给定无向图$G=(V,E)$，如果U属于V，并且对于任意u,v包含于U 有$< u,v >$包含于E，则称U是G的完全子图，G的完全子图U是G的团，当且仅当U不包含在G的更大的完全子图中，G的最大团是指G中所含顶点数目最多的团。如果U属于V，并且对于任意u,v包含于U有$< u,v >$不包含于E，则称U是G的空子图，G的空子图U是G的独立集，当且仅当U不包含在G的更大的独立集，G的最大团是指G中所含顶点数目最多的独立集。

一些性质： 最大独立集+最小覆盖集$=V$，最大团=补图的最大独立集，最小覆盖集=最大匹配

```
#include <cstdio>
bool am[100][100];
int ans;
int c[100];
int U[100][100];
int n;
bool dfs(int rest,int num)
{
    if (!rest)
    {
        if (num>=ans)
            return 1;
        else
            return 0;
    }
    int pre=-1;
    for (int i=0;i<rest && rest-i+num>=ans;i++)
    {
        int idx=U[num][i];
        if (num+c[idx]<ans)
            return 0;
        int nrest=0;
        for (int j=i+1; j<rest; j++)
            if (am[idx][U[num][j]])
                U[num+1][nrest++]=U[num][j];
        if (dfs(nrest,num+1))
            return 1;
    }
    return 0;
}
int main()
{
    while (scanf("%d",&n),n)
```

```
34      {
35          for (int i=0;i<n;i++)
36              for (int j=0;j<n;j++)
37                  scanf("%d",&am[i][j]);
38          ans=0;
39          for (int i=n-1; i>=0; i--)
40          {
41              int rest=0;
42              for (int j=i+1; j<n; j++)
43                  if (am[i][j])
44                      U[0][rest++]=j;
45              ans+=dfs(rest,0);
46              c[i]=ans;
47          }
48          printf("%d\n",ans);
49      }
50      return 0;
51  }
```

6.10 双连通分量

标号从0起

```
1   #include<cstdio>
2   #include<cstring>
3   #include<stack>
4   #include<queue>
5   #include<algorithm>
6   using namespace std;
7   const int MAXN=100000*2;
8   const int MAXM=200000;
9   struct edges
10  {
11      int to,next;
12      bool cut,visit;
13  } edge[MAXM<<1];
14  int head[MAXN],low[MAXN],dpt[MAXN],L;
15  bool visit[MAXN],cut[MAXN];
16  void init(int n)
17  {
18      L=0;
19      memset(head,-1,4*n);
20      memset(visit,0,n);
21  }
22  void add_edge(int u,int v)
23  {
24      edge[L].cut=edge[L].visit=0;
25      edge[L].to=v;
26      edge[L].next=head[u];
```

```
27      head[u]=L++;
28  }
29  int idx;
30  stack<int> st;
31  int bcc[MAXM];
32  void dfs(int u,int fu,int deg)
33  {
34      cut[u]=0;
35      visit[u]=1;
36      low[u]=dpt[u]=deg;
37      int tot=0;
38      for (int i=head[u]; i!=-1; i=edge[i].next)
39      {
40          int v=edge[i].to;
41          if (edge[i].visit)
42              continue;
43          st.push(i/2);
44          edge[i].visit=edge[i^1].visit=1;
45          if (visit[v])
46          {
47              low[u]=dpt[v]>low[u]?low[u]:dpt[v];
48              continue;
49          }
50          dfs(v,u,deg+1);
51          edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
52          if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
53          if (low[v]>=dpt[u] || u==fu)
54          {
55              while (st.top()!=i/2)
56              {
57                  int x=st.top()*2,y=st.top()*2+1;
58                  bcc[st.top()]=idx;
59                  st.pop();
60              }
61              bcc[i/2]=idx++;
62              st.pop();
63          }
64          low[u]=low[v]>low[u]?low[u]:low[v];
65          tot++;
66      }
67      if (u==fu && tot>1) cut[u]=1;
68  }
69  int main()
70  {
71      int n,m;
72      while (scanf("%d%d",&n,&m)!=EOF)
73      {
74          init(n);
```

```
75          for (int i=0; i<m; i++)
76          {
77              int u,v;
78              scanf("%d%d",&u,&v);
79              add_edge(u,v);
80              add_edge(v,u);
81          }
82          idx=0;
83          for (int i=0; i<n; i++)
84              if (!visit[i])
85                  dfs(i,i,0);
86      }
87      return 0;
88  }
```

6.11 割点与桥

```
1   #include<cstdio>
2   #include<cstring>
3   const int MAXN=10000;
4   struct edges
5   {
6       int to,next;
7       bool cut,visit;
8       int from;
9   } edge[MAXN-1<<1];
10  int head[MAXN],low[MAXN],dfn[MAXN],L;
11  bool visit[MAXN],cut[MAXN];
12  void init(int n)
13  {
14      L=0;
15      memset(head,-1,4*n);
16      memset(cut,0,4*n);
17      memset(visit,0,4*n);
18  }
19  void add_edge(int u,int v)
20  {
21      edge[L].from=u;
22      edge[L].cut=edge[L].visit=0;
23      edge[L].to=v;
24      edge[L].next=head[u];
25      head[u]=L++;
26  }
27  int idx;
28  void dfs(int u,int fu)
29  {
30      visit[u]=1;
31      low[u]=dfn[u]=idx++;
```

```
32      int tot=0;
33      for (int i=head[u]; i!=-1; i=edge[i].next)
34      {
35          int v=edge[i].to;
36          if (edge[i].visit)
37              continue;
38          edge[i].visit=edge[i^1].visit=1;
39          if (visit[v])
40          {
41              low[u]=dfn[v]>low[u]?low[u]:dfn[v];
42              continue;
43          }
44          dfs(v,u);
45          edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
46          if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
47          low[u]=low[v]>low[u]?low[u]:low[v];
48          tot++;
49      }
50      if (u==fu && tot>1) cut[u]=1;
51  }
52  int main()
53  {
54      int t;
55      scanf("%d",&t);
56      while (t--)
57      {
58          int n,m;
59          scanf("%d%d",&n,&m);
60          init(n);
61          for (int i=0; i<m; i++)
62          {
63              int u,v;
64              scanf("%d%d",&u,&v);
65              add_edge(--u,--v);
66              add_edge(v,u);
67          }
68          for (int i=0; i<n; i++)
69              if (!visit[i])
70              {
71                  idx=0;
72                  dfs(i,i);
73              }
74      }
75      return 0;
76  }
```

6.12 LCA

在线LCA，bfs

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
const int NSIZE = 50000;
const int DEG = 20;
struct trees
{

    int fa[DEG];
    int head,deg;
} tree[NSIZE];
struct edges
{
    int to , next;
} edge[NSIZE];
struct states
{
    int u,fu,deg;
};
int L;
void add_edge(int x, int y)
{
    edge[L].to = y;
    edge[L].next = tree[x].head;
    tree[x].head = L++;
}
int Root;
void BFS(int s)
{
    queue<states> que;
    states st;
    st.deg=0;
    st.fu=st.u=s;
    que.push(st);
    while(!que.empty())
    {
        states st=que.front();
        que.pop();
        tree[st.u].deg = st.deg;
        tree[st.u].fa[0] = st.fu;
        for (int i=1;i<DEG;i++)
            tree[st.u].fa[i]=s;
        for (int tmp=st.fu,num=1;tree[tmp].deg;tmp=tree[st.u].fa[
           num++])
            tree[st.u].fa[num]=tree[tmp].fa[num-1];
        for(int i = tree[st.u].head ; i != -1; i = edge[i].next)
        {
```

```
48              int v = edge[i].to;
49              if (v == st.fu) continue;
50              states nst;
51              nst.u=v;
52              nst.fu=st.u;
53              nst.deg=st.deg+1;
54              que.push(nst);
55          }
56      }
57  }
58  int LCA(int x, int y)
59  {
60      if(tree[x].deg > tree[y].deg) swap(x,y);
61      int hx=tree[x].deg,hy=tree[y].deg;
62      int tx=x,ty=y;
63      for (int det=hy-hx,i=0; det; det>>=1,i++)
64          if (det&1)
65              ty=tree[ty].fa[i];
66      if(tx == ty) return tx;
67      for (int i=DEG-1; i>=0; i--)
68      {
69          if(tree[tx].fa[i] == tree[ty].fa[i])
70              continue;
71          tx = tree[tx].fa[i];
72          ty = tree[ty].fa[i];
73      }
74      return tree[tx].fa[0];
75  }
76  int main()
77  {
78      int t;
79      scanf("%d",&t);
80      while(t--)
81      {
82          int n;
83          scanf("%d",&n);
84          L = 0;
85          for(int i = 0 ; i < n ; i++)
86              tree[i].head = -1;
87          for(int i = 0 ; i < n-1 ; i++)
88          {
89              int a,b;
90              scanf("%d%d",&a ,&b);
91              add_edge(a-1,b-1);
92              add_edge(b-1,a-1);
93          }
94          Root=0;
95          BFS(Root);
```

```
96          int a,b;
97          scanf("%d%d",&a,&b);
98          int lca=LCA(a-1,b-1)+1;
99          printf("%d\n",lca);
100     }
101     return 0;
102 }
```

6.13 最优比例生成树

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<math.h>
4  struct
5  {
6      int x,y;
7      double z;
8  } node[1100];
9  struct
10 {
11     double l,c;
12 } map[1100][1100];
13 int n,l,f[1100],pre[1100];
14 double dis[1100];
15 double mst(double x)
16 {
17     int i,j,tmp;
18     double min,s=0,t=0;
19     memset(f,0,sizeof(f));
20     f[1]=1;
21     for (i=2; i<=n; i++)
22     {
23         dis[i]=map[1][i].c-map[1][i].l*x;
24         pre[i]=1;
25     }
26     for (i=1; i<n; i++)
27     {
28         min=1e10;
29         for (j=1; j<=n; j++)
30             if (!f[j] && min>dis[j])
31             {
32                 min=dis[j];
33                 tmp=j;
34             }
35         f[tmp]=1;
36         t+=map[pre[tmp]][tmp].l;
37         s+=map[pre[tmp]][tmp].c;
38         for (j=1; j<=n; j++)
```

```
39              if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
40              {
41                  dis[j]=map[tmp][j].c-map[tmp][j].l*x;
42                  pre[j]=tmp;
43              }
44      }
45      return s/t;
46  }
47  int main()
48  {
49      int i,j;
50      double a,b;
51      scanf("%d",&n);
52      while (n)
53      {
54          for (i=1; i<=n; i++)
55              scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
56          for (i=1; i<=n; i++)
57              for (j=i+1; j<=n; j++)
58              {
59                  map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[
                       j].x)*(node[i].x-node[j].x)+(node[i].y-node[j
                       ].y)*(node[i].y-node[j].y));
60                  map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z)
                       ;
61              }
62          a=0,b=mst(a);
63          while (fabs(b-a)>1e-8)
64          {
65              a=b;
66              b=mst(a);
67          }
68          printf("%.3f\n",b);
69          scanf("%d",&n);
70      }
71  }
```

6.14 全局最小割

```
1   #include <iostream>
2   using namespace std;
3   const int maxn=510;
4   int map[maxn][maxn];
5   int n;
6   void contract(int x,int y)
7   {
8       int i,j;
9       for (i=0; i<n; i++)
10          if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
```

```
11      for (i=y+1; i<n; i++) for (j=0; j<n; j++)
12          {
13              map[i-1][j]=map[i][j];
14              map[j][i-1]=map[j][i];
15          }
16      n--;
17  }
18  int w[maxn],c[maxn];
19  int sx,tx;
20  int mincut()
21  {
22      int i,j,k,t;
23      memset(c,0,sizeof(c));
24      c[0]=1;
25      for (i=0; i<n; i++) w[i]=map[0][i];
26      for (i=1; i+1<n; i++)
27      {
28          t=k=-1;
29          for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
30                  k=w[t=j];
31          c[sx=t]=1;
32          for (j=0; j<n; j++) w[j]+=map[t][j];
33      }
34      for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
35  }
36  int main()
37  {
38      int i,j,k,m;
39      while (scanf("%d%d",&n,&m)!=EOF)
40      {
41          memset(map,0,sizeof(map));
42          while (m--)
43          {
44              scanf("%d%d%d",&i,&j,&k);
45              map[i][j]+=k;
46              map[j][i]+=k;
47          }
48          int mint=999999999;
49          while (n>1)
50          {
51              k=mincut();
52              if (k<mint) mint=k;
53              contract(sx,tx);
54          }
55          printf("%d\n",mint);
56      }
57      return 0;
58  }
```

6.15 欧拉路

6.15.1 有向图

```
void solve(int x)
{
    int i;
    if (!match[x])
    {
        path[++l]=x;
        return ;
    }
    for (i=1; i<=n; i++)
        if (b[x][i])
        {
            b[x][i]--;
            match[x]--;
            solve(i);
        }
    path[++l]=x;
}
```

6.15.2 无向图

```
void solve(int x)
{
    int i;
    if (!match[x])
    {
        path[++l]=x;
        return ;
    }
    for (i=1; i<=n; i++)
        if (b[x][i])
        {
            b[x][i]--;
            b[i][x]--;
            match[x]--;
            match[i]--;
            solve(i);
        }
    path[++l]=x;
}
```

6.15.3 混合图

zju1992

```
int in[MAXN+100],out[MAXN+100];
int main()
{
```

```
4       int t;
5       scanf("%d",&t);
6       while (t--)
7       {
8           int n,m;
9           scanf("%d%d",&n,&m);
10          N=n+2;L=-1;
11          for (int i=0;i<N;i++)
12              head[i]=-1;
13          memset(in,0,sizeof(in));
14          memset(out,0,sizeof(out));
15
16          for (int i=0;i<m;i++)
17          {
18              int x,y,z;
19              scanf("%d%d%d",&x,&y,&z);
20              in[y]++;out[x]++;
21              if (!z)
22                  add_edge(x,y,1);
23          }
24          int flag=1;
25          for (int i=1;i<=n;i++)
26          {
27              if (in[i]-out[i]>0)
28                  add_edge(i,n+1,(in[i]-out[i])/2);
29              else
30              if (out[i]-in[i]>0)
31                  add_edge(0,i,(out[i]-in[i])/2);
32              //printf("%d %d %d\n",i,out[i],in[i]);
33              if ((in[i]+out[i])&1)
34              {
35                  flag=0;
36                  break;
37              }
38          }
39          maxflow(0,n+1);
40          for (int i=head[0];i!=-1;i=edge[i].next)
41              if (edge[i].cap>0 && edge[i].cap>edge[i].flow)
42              {
43                  flag=0;
44                  break;
45              }
46          if (flag)
47              puts("possible");
48          else
49              puts("impossible");
50      }
51      return 0;
```

```
}
```

6.16 K短路

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
int K;
class states
{
public:
    int cost,id;
};
int dist[1000];
class cmp
{
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost>j.cost;
    }
};
class cmp2
{
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost+dist[i.id]>j.cost+dist[j.id];
    }
};
struct edges
{
    int to,next,cost;
} edger[100000],edge[100000];
int headr[1000],head[1000],Lr,L;
void dijkstra(int s)
{
    states u;
    u.id=s;
    u.cost=0;
    dist[s]=0;
    priority_queue<states,vector<states>,cmp> q;
    q.push(u);
    while (!q.empty())
    {
        u=q.top();
        q.pop();
```

```
45          if (u.cost!=dist[u.id]) continue;
46          for (int i=headr[u.id]; i!=-1; i=edger[i].next)
47          {
48              states v=u;
49              v.id=edger[i].to;
50              if (dist[v.id]>dist[u.id]+edger[i].cost)
51              {
52                  v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
53                  q.push(v);
54              }
55          }
56      }
57  }
58  int num[1000];
59  void init(int n)
60  {
61      Lr=L=0;
62      memset(head,-1,4*n);
63      memset(headr,-1,4*n);
64      memset(dist,63,4*n);
65      memset(num,0,4*n);
66  }
67  void add_edge(int u,int v,int x)
68  {
69      edge[L].to=v;
70      edge[L].cost=x;
71      edge[L].next=head[u];
72      head[u]=L++;
73      edger[Lr].to=u;
74      edger[Lr].cost=x;
75      edger[Lr].next=headr[v];
76      headr[v]=Lr++;
77  }
78  int a_star(int s,int t)
79  {
80      if (dist[s]==0x3f3f3f3f)
81          return -1;
82      priority_queue<states,vector<states>,cmp2> q;
83      states tmp;
84      tmp.id=s;
85      tmp.cost=0;
86      q.push(tmp);
87      while (!q.empty())
88      {
89          states u=q.top();
90          q.pop();
91          num[u.id]++;
92          if (num[t]==K)
```

```
93              return u.cost;
94          for (int i=head[u.id]; i!=-1; i=edge[i].next)
95          {
96              int v=edge[i].to;
97              tmp.id=v;
98              tmp.cost=u.cost+edge[i].cost;
99              q.push(tmp);
100         }
101     }
102     return -1;
103 }
104 int main()
105 {
106     int n,m;
107     scanf("%d%d",&n,&m);
108     init(n);
109     for (int i=0; i<m; i++)
110     {
111         int u,v,x;
112         scanf("%d%d%d",&u,&v,&x);
113         add_edge(u-1,v-1,x);
114     }
115     int s,t;
116     scanf("%d%d%d",&s,&t,&K);
117     if (s==t)
118         K++;
119     dijkstra(t-1);
120     printf("%d\n",a_star(s-1,t-1));
121 }
```

6.17 稳定婚姻

假定有n个男生和M个女生，理想的拍拖状态就是对于每对情侣(a,b)，找不到另一对情侣(c,d)使得c更喜欢b,b也更喜欢c，同理，对a来说也没有(e,f)使得a更喜欢e而e更喜欢a,当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。

求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如leokan向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为leokan的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢leokan高，这个女生也暂时成为leokan的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```
1 #include<string.h>
2 #include<stdio.h>
3 #define N 1050
4 int boy[N][N];
5 int girl[N][N];
6 int ans[N];
7 int cur[N];
8 int n;
```

```
9  void getMarry(int g)
10 {
11   for (int i=ans[g]+1;i<n;i++)
12   {
13     int b=girl[g][i]-1;
14     if (cur[b]<0)
15     {
16       ans[g]=i;
17       cur[b]=g;
18       return;
19     }
20     int og=cur[b];
21     if (boy[b][og] > boy[b][g])
22     {
23       cur[b]=g;
24       ans[g]=i;
25       getMarry(og);
26       return;
27     }
28   }
29 };
30 int main()
31 {
32   int t,a;
33   scanf("%d",&t);
34   while(t--)
35   {
36     memset(girl,0,sizeof(girl));
37     memset(boy,0,sizeof(boy));
38     scanf("%d",&n);
39     for (int i=0;i<n;i++)
40       for (int j=0;j<n;j++)
41         scanf("%d",&girl[i][j]);
42     for (int i=0;i<n;i++)
43       for (int j=0;j<n;j++)
44       {
45         scanf("%d",&a);
46         boy[i][a-1]=j;
47       }
48     memset(cur,0xff,sizeof(cur));
49     memset(ans,0xff,sizeof(ans));
50     for (int i=0;i<n;i++)
51       getMarry(i);
52     for (int i=0;i<n;i++)
53       printf("%d\n",girl[i][ans[i]]);
54   }
55   return 0;
56 }
```

6.18 最小树形图

```
const int inf = 19921005;
int n,m,u,v,cost,dis[1001][1001],L;

void init(int n)
{
    L = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            dis[i][j] = inf;
}

struct Edge
{
    int u,v,cost;
};

Edge e[1001*1001];

int pre[1001],id[1001],visit[1001],in[1001];

int zhuliu(int root,int n,int m,Edge e[])
{
    int res = 0,u,v;
    while (true)
    {
        for (int i = 0; i < n; i++)
            in[i] = inf;
        for (int i = 0; i < m; i++)
            if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
            {
                pre[e[i].v] = e[i].u;
                in[e[i].v] = e[i].cost;
            }
        for (int i = 0; i < n; i++)
            if (i != root)
                if (in[i] == inf)   return -1;
        int tn = 0;
        memset(id,-1,sizeof(id));
        memset(visit,-1,sizeof(visit));
        in[root] = 0;
        for (int i = 0; i < n; i++)
        {
            res += in[i];
            v = i;
            while (visit[v] != i && id[v] == -1 && v != root)
            {
```

```
47                  visit[v] = i;
48                  v = pre[v];
49              }
50              if(v != root && id[v] == -1)
51              {
52                  for(int u = pre[v] ; u != v ; u = pre[u])
53                      id[u] = tn;
54                  id[v] = tn++;
55              }
56          }
57          if(tn == 0)    break;
58          for (int i = 0; i < n; i++)
59              if (id[i] == -1)
60                  id[i] = tn++;
61          for (int i = 0; i < m;)
62          {
63              int v = e[i].v;
64              e[i].u = id[e[i].u];
65              e[i].v = id[e[i].v];
66              if (e[i].u != e[i].v)
67                  e[i++].cost -= in[v];
68              else
69                  swap(e[i],e[--m]);
70          }
71          n = tn;
72          root = id[root];
73      }
74      return res;
75  }
76
77  int main()
78  {
79      freopen("in.txt","r",stdin);
80      while (scanf("%d%d",&n,&m) != EOF)
81      {
82          init(n);
83          for (int i = 0; i < m; i++)
84          {
85              scanf("%d%d%d",&u,&v,&cost);
86              if (u == v) continue;
87              dis[u][v] = min(dis[u][v],cost);
88          }
89          L = 0;
90          for (int i = 0; i < n; i++)
91              for (int j = 0; j < n; j++)
92                  if (dis[i][j] != inf)
93                  {
94                      e[L].u = i;
```

```
95                      e[L].v = j;
96                      e[L++].cost = dis[i][j];
97                  }
98          printf("%d\n",zhuliu(0,n,L,e));
99      }
100     return 0;
101 }
```

7 计算几何

7.1 基本函数

7.1.1 Point定义

```
struct Point
{
    double x, y;
    Point() {}
    Point(double _x, double _y)
    {
        x = _x, y = _y;
    }
    Point operator -(const Point &b)const
    {
        return Point(x - b.x, y - b.y);
    }
    double operator *(const Point &b)const
    {
        return x * b.y - y * b.x;
    }
    double operator &(const Point &b)const
    {
        return x * b.x + y * b.y;
    }
    void transXY(double B)
    {
        double tx = x,ty = y;
        x = tx*cos(B) - ty*sin(B);
        y = tx*sin(B) + ty*cos(B);
    }
};
```

7.1.2 Line定义

```
struct Line
{
    Point s, e;
    double k;
    Line() {}
    Line(Point _s, Point _e)
    {
        s = _s, e = _e;
        k = atan2(e.y - s.y, e.x - s.x);
    }
    Point operator &(const Line &b)const
    {
        Point res = s;
```

```
14          //注意：有些题目可能会有直线相交或者重合情况
15          //可以把返回值改成pair<Point,int>来返回两直线的状态。
16          double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s -
               b.e));
17          res.x += (e.x - s.x) * t;
18          res.y += (e.y - s.y) * t;
19          return res;
20      }
21  };
```

7.1.3 距离：两点距离

```
1  double dist2(Point a, Point b)
2  {
3      return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
4  }
```

7.1.4 距离：点到直线距离

result:点到直线最近点

```
1  Point NPT(Point P, Line L)
2  {
3      Point result;
4      double a, b, t;
5
6      a = L.p2.x - L.p1.x;
7      b = L.p2.y - L.p1.y;
8      t = ( (P.x - L.p1.x) * a + (P.y - L.p1.y) * b ) / (a * a + b
          * b);
9
10     result.x = L.p1.x + a * t;
11     result.y = L.p1.y + b * t;
12     return dist2(P, result);
13 }
```

7.1.5 距离：点到线段距离

res：点到线段最近点

```
1  double dist2(Point p1, Point p2, Point p)
2  {
3      Point res;
4      double a, b, t;
5      a = p2.x - p1.x;
6      b = p2.y - p1.y;
7      t = ((p.x - p1.x) * a + (p.y - p1.y) * b) / (a * a + b * b);
8      if (t >= 0 && t <= 1)
9      {
10         res.x = p1.x + a * t;
11         res.y = p1.y + b * t;
```

```
12      }
13      else
14      {
15          if (dist2(p, p1) < dist2(p, p2))
16              res = p1;
17          else
18              res = p2;
19      }
20      return dist2(p, res);
21  }
```

旧版

```
1  double CalcDis(Point a,Point s,Point e) //点到线段距离
2  {
3      if (pmult(Point(s,e),Point(s,a)) < 0 || pmult(Point(e,s),
          Point(e,a)) < 0)
4          return min(CalcDis(a,s),CalcDis(a,e));
5      return abs(xmult(Point(a,s),Point(a,e)))/CalcDis(s,e);
6  }
```

7.1.6 面积：多边形

点按逆时针排序。

```
1  double CalcArea(Point p[], int n)
2  {
3      double res = 0;
4      for (int i = 0; i < n; i++)
5          res += (p[i] * p[(i + 1) % n]) / 2;
6      return res;
7  }
```

7.1.7 判断：线段相交

```
1  bool inter(Line l1,Line l2)
2  {
3      return (max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
4              max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
5              max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
6              max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
7              ((l2.s-l1.s)*(l1.e-l1.s))*((l2.e-l1.s)*(l1.e-l1.s))
                  <= 0 &&
8              ((l1.s-l2.s)*(l2.e-l2.s))*((l1.e-l2.s)*(l2.e-l2.s))
                  <= 0);
9  }
```

7.2 圆

7.2.1 面积：两圆相交

圆不可包含

```
double dis(int x,int y)
{
    return sqrt((double)(x*x+y*y));
}
double area(int x1,int y1,int x2,int y2,double r1,double r2)
{
    double s=dis(x2-x1,y2-y1);
    if(r1+r2<s) return 0;
    else if(r2-r1>s) return PI*r1*r1;
    else if(r1-r2>s) return PI*r2*r2;
    double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
    double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
    return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
}
```

7.2.2 三角形外接圆

```
void CircumscribedCircle()
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    tp = Point((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
    l[0] = Line(tp,Point(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x
       )));
    tp = Point((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
    l[1] = Line(tp,Point(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x
       )));
    tp = LineToLine(l[0],l[1]);
    r = Point(tp,p[0]).Length();
    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}
```

7.2.3 三角形内切圆

```
void InscribedCircle()
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    if (xmult(Point(p[0],p[1]),Point(p[0],p[2])) < 0)
        swap(p[1],p[2]);
    for (int i = 0; i < 3; i++)
        len[i] = Point(p[i],p[(i+1)%3]).Length();
    tr = (len[0]+len[1]+len[2])/2;
    r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
    for (int i = 0; i < 2; i++)
    {
        v = Point(p[i],p[i+1]);
        tv = Point(-v.y,v.x);
```

```
15          tr = tv.Length();
16          tv = Point(tv.x*r/tr,tv.y*r/tr);
17          tp = Point(p[i].x+tv.x,p[i].y+tv.y);
18          l[i].s = tp;
19          tp = Point(p[i+1].x+tv.x,p[i+1].y+tv.y);
20          l[i].e = tp;
21      }
22      tp = LineToLine(l[0],l[1]);
23      printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
24  }
```

7.2.4 点对圆的两个切点

```
1  void calc_qie(Point poi,Point o,double r,Point &result1,Point &
       result2) {
2      double line=sqrt((poi.x-o.x)*(poi.x-o.x)+(poi.y-o.y)*(poi.y-o
           .y));
3      double angle=acos(r/line);
4      Point unitvector,lin;
5      lin.x=poi.x-o.x;
6      lin.y=poi.y-o.y;
7      unitvector.x=lin.x/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
8      unitvector.y=lin.y/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
9      result1=Rotate(unitvector,-angle);
10     result2=Rotate(unitvector,angle);
11     result1.x+=o.x;
12     result1.y+=o.y;
13     result2.x+=o.x;
14     result2.y+=o.y;
15     return;
16 }
```

7.2.5 两圆公切点

```
1  void Gao()
2  {
3      tn = 0;
4      Point a,b,vab;
5      double tab,tt,dis,theta;
6      for (int i = 0; i < tc; i++)
7          for (int j = 0; j < tc; j++)
8              if (i != j)
9              {
10                 a = c[i];
11                 b = c[j];
12                 vab = Point(a,b);
13                 tab = atan2(vab.y,vab.x);
14                 dis = sqrt(vab.x*vab.x+vab.y*vab.y);
15                 if (b.r > a.r)
16                     tt = asin((b.r-a.r)/dis);
```

```
17                  else
18                      tt = -asin((a.r-b.r)/dis);
19                  theta = tab+pi/2+tt;
20                  tp[tn++] = Point(a.x+a.r*cos(theta),a.y+a.r*sin(
                        theta));
21                  tp[tn++] = Point(b.x+b.r*cos(theta),b.y+b.r*sin(
                        theta));
22              }
23  }
```

7.3 矩阵

7.3.1 基本矩阵

按向量$\overrightarrow{(x,y,z)}$平移:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

按比例(x,y,z)缩放:

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕向量$\overrightarrow{(x,y,z)}$旋转$angle$角度:

$$\begin{pmatrix} x^2\times(1-c)+c & x\times y\times(1-c)-z\times s & x\times z\times(1-c)+y\times s & 0 \\ y\times x\times(1-c)+z\times s & y^2\times(1-c)+c & y\times z\times(1-c)-x\times s & 0 \\ x\times z\times(1-c)-y\times s & y\times z\times(1-c)+x\times s & z^2\times(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s=sin(angle) \\ c=cos(angle) \end{cases}$$

7.3.2 刘汝佳的几何教室

```
1   const double pi = acos(-1.0);
2
3   int n,m,q;
4   struct Point
5   {
6       double a,b,c,d;
7   };
8   Point p[50000],f[50000];
9
10  double a,b,c,theta,mt[4][4],tmp[4][4],tmt[4][4],rmt[4][8];
11  char com[20];
12
13  void TRANSLATE()
```

```
14 {
15     memset(tmt,0,sizeof(tmt));
16     tmt[0][0] = tmt[1][1] = tmt[2][2] = tmt[3][3] = 1;
17     tmt[3][0] = a;
18     tmt[3][1] = b;
19     tmt[3][2] = c;
20     memset(tmp,0,sizeof(tmp));
21     for (int i = 0; i < 4; i++)
22         for (int j = 0; j < 4; j++)
23             for (int k = 0; k < 4; k++)
24                 tmp[i][j] += mt[i][k]*tmt[k][j];
25     for (int i = 0; i < 4; i++)
26         for (int j = 0; j < 4; j++)
27             mt[i][j] = tmp[i][j];
28 }
29
30 void ROTATE()
31 {
32     theta = -theta*pi/180;
33     memset(tmt,0,sizeof(tmt));
34     tmt[3][3] = 1;
35     tmt[0][0] = cos(theta)+(1-cos(theta))*a*a;
36     tmt[1][0] = (1-cos(theta))*a*b+c*sin(theta);
37     tmt[2][0] = (1-cos(theta))*a*c-b*sin(theta);
38     tmt[0][1] = (1-cos(theta))*a*b-c*sin(theta);
39     tmt[1][1] = cos(theta)+(1-cos(theta))*b*b;
40     tmt[2][1] = (1-cos(theta))*b*c+a*sin(theta);
41     tmt[0][2] = (1-cos(theta))*a*c+b*sin(theta);
42     tmt[1][2] = (1-cos(theta))*b*c-a*sin(theta);
43     tmt[2][2] = cos(theta)+(1-cos(theta))*c*c;
44     memset(tmp,0,sizeof(tmp));
45     for (int i = 0; i < 4; i++)
46         for (int j = 0; j < 4; j++)
47             for (int k = 0; k < 4; k++)
48                 tmp[i][j] += mt[i][k]*tmt[k][j];
49     for (int i = 0; i < 4; i++)
50         for (int j = 0; j < 4; j++)
51             mt[i][j] = tmp[i][j];
52 }
53
54 void SCALE()
55 {
56     memset(tmt,0,sizeof(tmt));
57     tmt[0][0] = a;
58     tmt[1][1] = b;
59     tmt[2][2] = c;
60     tmt[3][3] = 1;
61     memset(tmp,0,sizeof(tmp));
```

```
62      for (int i = 0; i < 4; i++)
63          for (int j = 0; j < 4; j++)
64              for (int k = 0; k < 4; k++)
65                  tmp[i][j] += mt[i][k]*tmt[k][j];
66      for (int i = 0; i < 4; i++)
67          for (int j = 0; j < 4; j++)
68              mt[i][j] = tmp[i][j];
69  }
70
71  void solvep(Point p)
72  {
73      memset(tmt,0,sizeof(tmt));
74      tmt[0][0] = p.a;
75      tmt[0][1] = p.b;
76      tmt[0][2] = p.c;
77      tmt[0][3] = 1;
78      memset(tmp,0,sizeof(tmp));
79      for (int i = 0; i < 1; i++)
80          for (int j = 0; j < 4; j++)
81              for (int k = 0; k < 4; k++)
82                  tmp[i][j] += tmt[i][k]*mt[k][j];
83      printf("%.2f␣%.2f␣%.2f\n",tmp[0][0],tmp[0][1],tmp[0][2]);
84  }
85
86  void solvef(Point f)
87  {
88      memset(tmt,0,sizeof(tmt));
89      tmt[0][0] = f.a;
90      tmt[1][0] = f.b;
91      tmt[2][0] = f.c;
92      tmt[3][0] = 0;
93      memset(tmp,0,sizeof(tmp));
94      for (int i = 0;i < 4;i++)
95          for (int j = 0;j < 1;j++)
96              for (int k = 0;k < 4;k++)
97                  tmp[i][j] += mt[i][k]*tmt[k][j];
98      tmp[3][0] += f.d;
99      double kk = tmp[0][0]*tmp[0][0]+tmp[1][0]*tmp[1][0]+tmp
           [2][0]*tmp[2][0];
100     kk = sqrt(1/kk);
101     for (int i = 0;i < 4;i++)
102         printf("%.2f␣",tmp[i][0]*kk);
103     printf("\n");
104 }
105
106 void solvermt()
107 {
108     memset(rmt,0,sizeof(rmt));
```

```
109     for (int i = 0;i < 4;i++)
110         for (int j = 0;j < 4;j++)
111             rmt[i][j] = mt[i][j];
112     rmt[0][4] = rmt[1][5] = rmt[2][6] = rmt[3][7] = 1;
113     for (int i = 0;i < 4;i++)
114     {
115         for (int j = i;j < 4;j++)
116             if (fabs(rmt[j][i]) > 1e-8)
117             {
118                 for (int k = i;k < 8;k++)
119                     swap(rmt[i][k],rmt[j][k]);
120                 break;
121             }
122         double tt = rmt[i][i];
123         for (int j = i;j < 8;j++)
124             rmt[i][j] /= tt;
125         for (int j = 0;j < 4;j++)
126             if (i != j)
127             {
128                 tt = rmt[j][i];
129                 for (int k = i;k < 8;k++)
130                     rmt[j][k] -= rmt[i][k]*tt;
131             }
132     }
133     for (int i = 0;i < 4;i++)
134         for (int j = 0;j < 4;j++)
135             mt[i][j] = rmt[i][4+j];
136 }
137
138 int main()
139 {
140     scanf("%d%d%d",&n,&m,&q);
141     for (int i = 0; i < n; i++)
142         scanf("%lf%lf%lf",&p[i].a,&p[i].b,&p[i].c);
143     for (int i = 0; i < m; i++)
144         scanf("%lf%lf%lf%lf",&f[i].a,&f[i].b,&f[i].c,&f[i].d);
145     memset(mt,0,sizeof(mt));
146     mt[0][0] = mt[1][1] = mt[2][2] = mt[3][3] = 1;
147     for (int i = 0; i < q; i++)
148     {
149         scanf("%s",com);
150         if (strcmp(com,"TRANSLATE") == 0)
151         {
152             scanf("%lf%lf%lf",&a,&b,&c);
153             TRANSLATE();
154         }
155         else if (strcmp(com,"ROTATE") == 0)
156         {
```

```
157             scanf("%lf%lf%lf%lf",&a,&b,&c,&theta);
158             ROTATE();
159         }
160         else if (strcmp(com,"SCALE") == 0)
161         {
162             scanf("%lf%lf%lf",&a,&b,&c);
163             SCALE();
164         }
165     }
166   //处理点
167     for (int i = 0; i < n; i++)
168         solvep(p[i]);
169   //处理面
170     solvermt();
171     for (int i = 0; i < m; i++)
172         solvef(f[i]);
173     return 0;
174 }
```

7.4 重心

```
1  Point CenterOfPolygon(Point poly[],int n)
2  {
3      Point p, p0, p1, p2, p3;
4      double m, m0;
5      p1 = poly[0];
6      p2 = poly[1];
7      p.x = p.y = m = 0;
8      for (int i = 2; i < n; i++)
9      {
10   p3 = poly[i];
11   p0.x = (p1.x + p2.x + p3.x) / 3.0;
12   p0.y = (p1.y + p2.y + p3.y) / 3.0;
13   m0 = p1.x * p2.y + p2.x * p3.y + p3.x * p1.y - p1.y * p2.x - p2
        .y * p3.x - p3.y * p1.x;
14   if (cmp(m + m0,0.0) == 0)
15       m0 += eps;
16   p.x = (m * p.x + m0 * p0.x) / (m + m0);
17   p.y = (m * p.y + m0 * p0.y) / (m + m0);
18   m = m + m0;
19   p2 = p3;
20     }
21     return p;
22 }
```

7.5 KD树

查找某个点距离最近的点，基本思想是每次分治把点分成两部分，建议按照坐标规模决定是垂直划分还是水平划分，查找时先往分到的那一部分查找，然后根据当前最优答案决定是否去另一个区间查找。

```
bool Div[MaxN];
void BuildKD(int deep,int l, int r, Point p[])\\记得备份一下P
{
    if (l > r) return;
    int mid = l + r >> 1;
    int minX, minY, maxX, maxY;
    minX = min_element(p + l, p + r + 1, cmpX)->x;
    minY = min_element(p + l, p + r + 1, cmpY)->y;
    maxX = max_element(p + l, p + r + 1, cmpX)->x;
    maxY = max_element(p + l, p + r + 1, cmpY)->y;
    Div[mid] = (maxX - minX >= maxY - minY);
    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY
       );
    BuildKD(l, mid - 1, p);
    BuildKD(mid + 1, r, p);
}

long long res;
void Find(int l, int r, Point a, Point p[])\\查找
{
    if (l > r)  return;
    int mid = l + r >> 1;
    long long dist = dist2(a, p[mid]);
    if (dist > 0)//如果有重点不能这样判断
        res = min(res, dist);
    long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
    int l1, l2, r1, r2;
    l1 = l, l2 = mid + 1;
    r1 = mid - 1, r2 = r;
    if (d > 0)
        swap(l1, l2), swap(r1, r2);
    Find(l1, r1, a, p);
    if (d * d < res)
        Find(l2, r2, a, p);
}
```

7.5.1 例题

查询一个点为中心的给定正方形内所有点并删除（2012金华网赛A）

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <queue>
using namespace std;

```

```
9  const int MaxN = 100000;
10 struct Point
11 {
12   int x,y,r;
13   int id;
14   bool del;
15 };
16
17 int cmpTyp;
18 bool cmp(const Point& a,const Point& b)
19 {
20   if (cmpTyp == 0)
21     return a.x < b.x;
22   else
23     return a.y < b.y;
24 }
25
26 int cnt[MaxN];
27 bool Div[MaxN];
28 int minX[MaxN],minY[MaxN],maxX[MaxN],maxY[MaxN];
29 void BuildKD(int l,int r,Point p[])
30 {
31   if (l > r)  return;
32   int mid = l+r>>1;
33   cmpTyp = 0;
34   minX[mid] = min_element(p+l,p+r+1,cmp)->x;
35   maxX[mid] = max_element(p+l,p+r+1,cmp)->x;
36   cmpTyp = 1;
37   minY[mid] = min_element(p+l,p+r+1,cmp)->y;
38   maxY[mid] = max_element(p+l,p+r+1,cmp)->y;
39
40   cnt[mid] = r-l+1;
41   cmpTyp = Div[mid] = (maxX[mid]-minX[mid] < maxY[mid]-minY[mid])
       ;
42   nth_element(p+l,p+mid,p+r+1,cmp);
43   BuildKD(l,mid-1,p);
44   BuildKD(mid+1,r,p);
45 }
46
47 queue<int> Q;
48 int Find(int l,int r,Point a,Point p[])
49 {
50   if (l > r)  return 0;
51   int mid = l+r>>1;
52   if (cnt[mid] == 0)  return 0;
53
54   if (maxX[mid] < a.x-a.r ||
55       minX[mid] > a.x+a.r ||
```

```
56        maxY[mid] < a.y-a.r ||
57        minY[mid] > a.y+a.r)
58      return 0;
59
60    int totdel = 0;
61
62    if (p[mid].del == false)
63      if (abs(p[mid].x-a.x) <= a.r && abs(p[mid].y-a.y) <= a.r)
64      {
65        p[mid].del = true;
66        Q.push(p[mid].id);
67        totdel++;
68      }
69
70    totdel += Find(l,mid-1,a,p);
71    totdel += Find(mid+1,r,a,p);
72
73    cnt[mid] -= totdel;
74
75    return totdel;
76  }
77
78  Point p[MaxN],tp[MaxN];
79  int n;
80
81  int main()
82  {
83    int cas = 1;
84    while (true)
85    {
86      scanf("%d",&n);
87      if (n == 0) break;
88
89      for (int i = 0;i < n;i++)
90      {
91        p[i].id = i;
92        int tx,ty;
93        scanf("%d%d%d",&tx,&ty,&p[i].r);
94        p[i].x = tx-ty;
95        p[i].y = tx+ty;
96        p[i].del = false;
97        tp[i] = p[i];
98      }
99      BuildKD(0,n-1,tp);
100
101     printf("Case␣#%d:\n",cas++);
102     int q;
103     scanf("%d",&q);
```

```
104     for (int i = 0;i < q;i++)
105     {
106       int id;
107       scanf("%d",&id);
108       int res = 0;
109       id--;
110       Q.push(id);
111       while (!Q.empty())
112       {
113         int now = Q.front();
114         Q.pop();
115         if (p[now].del == true) continue;
116         p[now].del = true;
117         res += Find(0,n-1,p[now],tp);
118       }
119       printf("%d\n",res);
120     }
121   }
122   return 0;
123 }
```

7.6 半平面交

直线左边代表有效区域。

```
1  bool HPIcmp(Line a, Line b)
2  {
3      if (fabs(a.k - b.k) > eps)    return a.k < b.k;
4      return ((a.s - b.s) * (b.e-b.s)) < 0;
5  }
6
7  Line Q[100];
8  void HPI(Line line[], int n, Point res[], int &resn)
9  {
10     int tot = n;
11     sort(line, line + n, HPIcmp);
12     tot = 1;
13     for (int i = 1; i < n; i++)
14         if (fabs(line[i].k - line[i - 1].k) > eps)
15             line[tot++] = line[i];
16     int head = 0, tail = 1;
17     Q[0] = line[0];
18     Q[1] = line[1];
19     resn = 0;
20     for (int i = 2; i < tot; i++)
21     {
22         if (fabs((Q[tail].e-Q[tail].s) * (Q[tail - 1].e-Q[tail -
               1].s)) < eps ||
```

```
23                  fabs((Q[head].e-Q[head].s) * (Q[head + 1].e-Q[
                        head + 1].s)) < eps)
24              return;
25          while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s
                ) * (line[i].e-line[i].s)) > eps)
26              tail--;
27          while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s
                ) * (line[i].e-line[i].s)) > eps)
28              head++;
29          Q[++tail] = line[i];
30      }
31      while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) *
            (Q[head].e-Q[head].s)) > eps)
32          tail--;
33      while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) *
            (Q[tail].e-Q[tail].s)) > eps)
34          head++;
35      if (tail <= head + 1) return;
36      for (int i = head; i < tail; i++)
37          res[resn++] = Q[i] & Q[i + 1];
38      if (head < tail + 1)
39          res[resn++] = Q[head] & Q[tail];
40  }
```

7.7 凸包

得到的凸包按照逆时针方向排序。

```
1   bool GScmp(Point a, Point b)
2   {
3       if (fabs(a.x - b.x) < eps)
4           return a.y < b.y - eps;
5       return a.x < b.x - eps;
6   }
7
8   void GS(Point p[], int n, Point res[], int &resn)
9   {
10      resn = 0;
11      int top = 0;
12      sort(p, p + n, GScmp);
13      for (int i = 0; i < n;)
14          if (resn < 2 || (res[resn - 1] - res[resn - 2]) * (p[i] -
                res[resn - 1]) > eps)
15              res[resn++] = p[i++];
16          else
17              --resn;
18      top = resn - 1;
19      for (int i = n - 2; i >= 0;)
```

```
20          if (resn < top + 2 || (res[resn - 1] - res[resn - 2]) * (
               p[i] - res[resn - 1]) > eps)
21              res[resn++] = p[i--];
22          else
23              --resn;
24      resn--;
25      if (resn < 3)   resn = 0;
26  }
```

7.8 直线与凸包求交点

复杂度$O(\log n)$。
需要先预处理几个东西。

```
1   //二分[la,lb]这段区间那条边与line相交
2   int Gao(int la,int lb,Line line)
3   {
4       if (la > lb)
5           lb += n;
6       int l = la,r = lb,mid;
7       while (l < r)
8       {
9           mid = l+r+1>>1;
10          if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-
               line.s)*(p[mid]-line.s),0) >= 0)
11              l = mid;
12          else
13              r = mid-1;
14      }
15      return l%n;
16  }
17  //求l与凸包的交点
18
19  //先调用Gettheta预处理出凸包每条边的斜率，然后处理成升序排列
20  double theta[maxn];
21
22  void Gettheta()
23  {
24      for (int i = 0;i < n;i++)
25      {
26          Point v = p[(i+1)%n]-p[i];
27          theta[i] = atan2(v.y,v.x);
28      }
29      for (int i = 1;i < n;i++)
30          if (theta[i-1] > theta[i]+eps)
31              theta[i] += 2*pi;
32  }
33
```

```
34 double Calc(Line l)
35 {
36     double tnow;
37     Point v = l.e-l.s;
38     tnow = atan2(v.y,v.x);
39     if (cmp(tnow,theta[0]) < 0)    tnow += 2*pi;
40     int pl = lower_bound(theta,theta+n,tnow)-theta;
41     tnow = atan2(-v.y,-v.x);
42     if (cmp(tnow,theta[0]) < 0)    tnow += 2*pi;
43     int pr = lower_bound(theta,theta+n,tnow)-theta;
44     //pl和pr是在l方向上距离最远的点对
45     pl = pl%n;
46     pr = pr%n;
47
48     if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
49         return 0.0;
50
51     int xa = Gao(pl,pr,l);
52     int xb = Gao(pr,pl,l);
53
54     if (xa > xb)    swap(xa,xb);
55     //与[xa,xa+1]和[xb,xb+1]这两条线段相交
56
57     if (cmp(v*(p[xa+1]-p[xa]),0) == 0)  return 0.0;
58     if (cmp(v*(p[xb+1]-p[xb]),0) == 0)  return 0.0;
59
60     Point pa,pb;
61     pa = Line(p[xa],p[xa+1])&l;
62     pb = Line(p[xb],p[xb+1])&l;
63     //题目：求直线切凸包得到的两部分的面积
64     double area0 = sum[xb]-sum[xa+1]+(pa*p[xa+1])/2.0+(p[xb]*pb)
          /2.0+(pb*pa)/2.0;
65     double area1 = sum[xa+n]-sum[xb+1]+(pb*p[xb+1])/2.0+(p[xa]*pa
          )/2.0+(pa*pb)/2.0;
66
67     return min(area0,area1);
68 }
```

7.9 三维凸包

暴力写法

```
1 #define eps 1e-7
2 #define MAXV 505
3
4 struct pt
5 {
6     double x, y, z;
```

```
7       pt() {}
8       pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
9       pt operator - (const pt p1)
10      {
11          return pt(x - p1.x, y - p1.y, z - p1.z);
12      }
13      pt operator * (pt p)
14      {
15          return pt(y*p.z-z*p.y, z*p.x-x*p.z, x*p.y-y*p.x);
16      }
17      double operator ^ (pt p)
18      {
19          return x*p.x+y*p.y+z*p.z;
20      }
21  };
22  struct _3DCH
23  {
24      struct fac
25      {
26          int a, b, c;
27          bool ok;
28      };
29      int n;
30      pt P[MAXV];
31      int cnt;
32      fac F[MAXV*8];
33      int to[MAXV][MAXV];
34      double vlen(pt a)
35      {
36          return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
37      }
38      double area(pt a, pt b, pt c)
39      {
40          return vlen((b-a)*(c-a));
41      }
42      double volume(pt a, pt b, pt c, pt d)
43      {
44          return (b-a)*(c-a)^(d-a);
45      }
46      double ptof(pt &p, fac &f)
47      {
48          pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
49          return (m * n) ^ t;
50      }
51      void deal(int p, int a, int b)
52      {
53          int f = to[a][b];
54          fac add;
```

```
55          if (F[f].ok)
56          {
57              if (ptof(P[p], F[f]) > eps)
58                  dfs(p, f);
59              else
60              {
61                  add.a = b, add.b = a, add.c = p, add.ok = 1;
62                  to[p][b] = to[a][p] = to[b][a] = cnt;
63                  F[cnt++] = add;
64              }
65          }
66      }
67      void dfs(int p, int cur)
68      {
69          F[cur].ok = 0;
70          deal(p, F[cur].b, F[cur].a);
71          deal(p, F[cur].c, F[cur].b);
72          deal(p, F[cur].a, F[cur].c);
73      }
74      bool same(int s, int t)
75      {
76          pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
77          return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(
                volume(a, b, c,
78                  P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].
                    c])) < eps;
79      }
80      void construct()
81      {
82          cnt = 0;
83          if (n < 4)
84              return;
85          bool sb = 1;
86          for (int i = 1; i < n; i++)
87          {
88              if (vlen(P[0] - P[i]) > eps)
89              {
90                  swap(P[1], P[i]);
91                  sb = 0;
92                  break;
93              }
94          }
95          if (sb)return;
96          sb = 1;
97          for (int i = 2; i < n; i++)
98          {
99              if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
100             {
```

```
101                 swap(P[2], P[i]);
102                 sb = 0;
103                 break;
104             }
105         }
106         if (sb)return;
107         sb = 1;
108         for (int i = 3; i < n; i++)
109         {
110             if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i
                    ])) > eps)
111             {
112                 swap(P[3], P[i]);
113                 sb = 0;
114                 break;
115             }
116         }
117         if (sb)return;
118         fac add;
119         for (int i = 0; i < 4; i++)
120         {
121             add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4,
                    add.ok = 1;
122             if (ptof(P[i], add) > 0)
123                 swap(add.b, add.c);
124             to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a
                    ] = cnt;
125             F[cnt++] = add;
126         }
127         for (int i = 4; i < n; i++)
128         {
129             for (int j = 0; j < cnt; j++)
130             {
131                 if (F[j].ok && ptof(P[i], F[j]) > eps)
132                 {
133                     dfs(i, j);
134                     break;
135                 }
136             }
137         }
138         int tmp = cnt;
139         cnt = 0;
140         for (int i = 0; i < tmp; i++)
141         {
142             if (F[i].ok)
143             {
144                 F[cnt++] = F[i];
145             }
```

```
146         }
147     }
148 //表面积
149     double area()
150     {
151         double ret = 0.0;
152         for (int i = 0; i < cnt; i++)
153         {
154             ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
155         }
156         return ret / 2.0;
157     }
158 //体积
159     double volume()
160     {
161         pt O(0, 0, 0);
162         double ret = 0.0;
163         for (int i = 0; i < cnt; i++)
164         {
165             ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
166         }
167         return fabs(ret / 6.0);
168     }
169 //表面三角形数
170     int facetCnt_tri()
171     {
172         return cnt;
173     }
174 //表面多边形数
175     int facetCnt()
176     {
177         int ans = 0;
178         for (int i = 0; i < cnt; i++)
179         {
180             bool nb = 1;
181             for (int j = 0; j < i; j++)
182             {
183                 if (same(i, j))
184                 {
185                     nb = 0;
186                     break;
187                 }
188             }
189             ans += nb;
190         }
191         return ans;
192     }
193
```

```
194     pt Fc[MAXV*8];
195     double V[MAXV*8];
196     pt Center()//重心
197     {
198         pt O(0,0,0);
199         for (int i = 0; i < cnt; i++)
200         {
201             Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)
                    /4.0;
202             Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)
                    /4.0;
203             Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)
                    /4.0;
204             V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
205         }
206         pt res = Fc[0],tmp;
207         double m = V[0];
208         for (int i = 1; i < cnt; i++)
209         {
210             if (fabs(m+V[i]) < eps)
211                 V[i] += eps;
212             tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
213             tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
214             tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
215             m += V[i];
216             res = tmp;
217         }
218         return res;
219     }
220 };
221
222 _3DCH hull;
223
224 int main()
225 {
226     while (scanf("%d",&hull.n) != EOF)
227     {
228         for (int i = 0; i < hull.n; i++)
229             scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i
                    ].z);
230         hull.construct();
231     }
232     return 0;
233 }
```

7.10 旋转卡壳

“对踵”

7.10.1 单个凸包

```
void solve(Point p[],int n)
{
    Point v;
    int cur = 1;
    for (int i = 0;i < n;i++)
    {
        v = p[i]-p[(i+1)%n];
        while (v*(p[(cur+1)%n]-p[cur]) < 0)
            cur = (cur+1)%n;
        //p[cur] -> p[i]
        //p[cur] -> p[i+1]
        //p[cur] -> (p[i],p[i+1])
    }
}
```

7.10.2 两个凸包

注意初始点的选取，代码只是个示例。
有时候答案需要取solve(p0,n,p1,m)和solve(p1,m,p0,n)的最优值。
何老鱼说我的是错的。。

```
void solve(Point p0[],int n,Point p1[],int m)
{
    Point v;
    int cur = 0;
    for (int i = 0;i < n;i++)
    {
        v = p0[i]-p0[(i+1)%n];
        while (v*(p1[(cur+1)%m]-p1[cur]) < 0)
            cur = (cur+1)%m;
        //p1[cur] -> p0[i]
        //p1[cur] -> p0[i+1]
        //p1[cur] -> (p0[i],p0[i+1])
    }
}
```

7.10.3 外接矩形

```
void solve()
{
    resa = resb = 1e100;
    double dis1,dis2;
```

```
5       Point xp[4];
6       Line l[4];
7       int a,b,c,d;
8       int sa,sb,sc,sd;
9       a = b = c = d = 0;
10      sa = sb = sc = sd = 0;
11      Point va,vb,vc,vd;
12      for (a = 0; a < n; a++)
13      {
14          va = Point(p[a],p[(a+1)%n]);
15          vc = Point(-va.x,-va.y);
16          vb = Point(-va.y,va.x);
17          vd = Point(-vb.x,-vb.y);
18          if (sb < sa)
19          {
20              b = a;
21              sb = sa;
22          }
23          while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
24          {
25              b = (b+1)%n;
26              sb++;
27          }
28          if (sc < sb)
29          {
30              c = b;
31              sc = sb;
32          }
33          while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
34          {
35              c = (c+1)%n;
36              sc++;
37          }
38          if (sd < sc)
39          {
40              d = c;
41              sd = sc;
42          }
43          while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
44          {
45              d = (d+1)%n;
46              sd++;
47          }
48
49          //卡在p[a],p[b],p[c],p[d]上
50          sa++;
51      }
52  }
```

7.11 三角形内点个数

7.11.1 无三点共线

```
Point p[1000], tp[2000], base;

bool cmp(const Point &a, const Point &b)
{
  return a.theta < b.theta;
}

int cnt[1000][1000];
int cntleft[1000][1000];
int n, m;

int calc(int a, int b, int c)
{
    Point p1 = p[b] - p[a], p2 = p[c] - p[a];
    if (atan2(p1.y, p1.x) > atan2(p2.y, p2.x))
        swap(b, c);
    if ((p[b] - p[a]) * (p[c] - p[a]) > 0)
        return cnt[a][c] - cnt[a][b] - 1;
    else
        return n - 3 - (cnt[a][c] - cnt[a][b] - 1);
}

int main(int argc, char const *argv[])
{
    int totcas;
    scanf("%d", &totcas);
    for (int cas = 1; cas <= totcas; ++cas)
    {
        scanf("%d", &n);
        for (int i = 0; i < n; ++i)
        {
            scanf("%lld%lld", &p[i].x, &p[i].y);
            p[i].id = i;
        }
        for (int i = 0; i < n; ++i)
        {
            m = 0;
            base = p[i];
            for (int j = 0; j < n; ++j)
                if (i != j)
                {
                    tp[m] = p[j];
                    Point v = tp[m]-base;
                    tp[m++].theta = atan2(v.y,v.x);
                }
```

```
46
47              sort(tp, tp + m, cmp);
48              for (int j = 0; j < m; ++j)
49                  tp[m + j] = tp[j];
50
51              //calc cnt
52              for (int j = 0; j < m; ++j)
53                  cnt[i][tp[j].id] = j;
54
55              //calc cntleft
56              for (int j = 0, k = 0, tot = 0; j < m; ++j)
57              {
58                  while (k == j || (k < j + m && (tp[j] - base) * (
                       tp[k] - base) > 0))
59                      k++, tot++;
60                  cntleft[i][tp[j].id] = --tot;
61              }
62          }
63
64          printf("Case␣%d:\n", cas);
65          int q;
66          scanf("%d", &q);
67          for (int i = 0; i < q; ++i)
68          {
69              int x, y, z;
70              scanf("%d%d%d", &x, &y, &z);
71              if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
72                  swap(y, z);
73              int res = cntleft[x][z] + cntleft[z][y] + cntleft[y][
                   x];
74              res += calc(x, y, z) + calc(y, z, x) + calc(z, x, y);
75              res -= 2 * (n - 3);
76              printf("%d\n", res);
77          }
78      }
79      return 0;
80  }
```

7.11.2 有三点共线且点有类别之分

```
1  int n,n0,n1,m;
2  Point p[3000], tp[3000], base;
3
4  bool cmp(const Point &a, const Point &b)
5  {
6      if ((a-base)*(b-base) == 0)
7      {
8          return (a-base).getMol() < (b-base).getMol();
9      }
```

```
10      return a.theta < b.theta;
11  }
12
13  int cnt[100][100];
14  int cntleft[100][100];
15
16  int calc(int a,int b,int c)
17  {
18      Point p1 = p[b]-p[a],p2 = p[c]-p[a];
19      if (atan2(1.0*p1.y,1.0*p1.x) > atan2(1.0*p2.y,1.0*p2.x))
20          swap(b,c);
21      int res = cnt[a][c]-cnt[a][b];
22      if ((p[b]-p[a])*(p[c]-p[a]) > 0)
23          return res;
24      else
25          return n1-res;
26  }
27
28  int main()
29  {
30      int cas = 0;
31      while (scanf("%d%d",&n0,&n1) != EOF)
32      {
33          n = n1+n0;
34          for (int i = 0; i < n; i++)
35          {
36              scanf("%I64d%I64d",&p[i].x,&p[i].y);
37              p[i].id = i;
38          }
39          for (int i = 0; i < n0; ++i)
40          {
41              m = 0;
42              base = p[i];
43              for (int j = 0; j < n; ++j)
44                  if (i != j)
45                  {
46                      tp[m] = p[j];
47                      Point v = tp[m]-base;
48                      tp[m++].theta = atan2(1.0*v.y,1.0*v.x);
49                  }
50
51              sort(tp, tp + m, cmp);
52              for (int j = 0; j < m; ++j)
53                  tp[m + j] = tp[j];
54
55              for (int j = 0,tot = 0; j < m; ++j)
56              {
57                  if (tp[j].id < n0)
```

```
58                      cnt[i][tp[j].id] = tot;
59                  else
60                      tot++;
61              }
62
63              for (int j = 0, k = 0, tot = 0; j < m; ++j)
64              {
65                  while (k == j || (k < j + m && (tp[j] - base) * (
                       tp[k] - base) > 0))
66                  {
67                      if (tp[k].id >= n0)
68                          tot++;
69                      k++;
70                  }
71                  if (tp[j].id >= n0)
72                      tot--;
73                  else
74                      cntleft[i][tp[j].id] = tot;
75              }
76          }
77
78          int ans = 0;
79          for (int i = 0; i < n0; i++)
80              for (int j = i+1; j < n0; j++)
81                  for (int k = j+1; k < n0; k++)
82                  {
83                      int x = i,y = j,z = k;
84
85                      if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
86                          swap(y, z);
87                      int res = cntleft[x][z] + cntleft[z][y] +
                           cntleft[y][x];
88
89                      res += calc(x, y, z) + calc(y, z, x) + calc(z
                           , x, y);
90
91                      res -= 2 * n1;
92
93                      //printf("%d %d %d %d\n",x,y,z,res);
94
95                      if (res%2 == 1)
96                          ans++;
97                  }
98          printf("Case␣%d:␣%d\n",++cas,ans);
99      }
100     return 0;
101 }
```

7.12 最近点对

7.12.1 类快排算法

```
double calc_dis(Point &a ,Point &b) {
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
//别忘了排序
bool operator<(const Point &a ,const Point &b) {
    if(a.y != b.y) return a.x < b.x;
    return a.x < b.x;
}
double Gao(int l ,int r ,Point pnts[]) {
    double ret = inf;
    if(l == r) return ret;
    if(l+1 ==r) {
        ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
        return ret;
    }
    if(l+2 ==r) {
        ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
        ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
        ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
        return ret;
    }

    int mid = l+r>>1;
    ret = min (ret ,Gao(l ,mid,pnts));
    ret = min (ret , Gao(mid+1, r,pnts));

    for(int c = l ; c<=r; c++)
        for(int d = c+1; d <=c+7 && d<=r; d++) {
            ret = min(ret , calc_dis(pnts[c],pnts[d]));
        }
    return ret;
}
```

7.12.2 随机增量法

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <map>
#include <vector>
#include <cmath>
#include <algorithm>
#define Point pair<double,double>
using namespace std;

```

```
11 const int step[9][2] =
      {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
12 int n,x,y,nx,ny;
13 map<pair<int,int>,vector<Point > > g;
14 vector<Point > tmp;
15 Point p[20000];
16 double tx,ty,ans,nowans;
17 vector<Point >::iterator it,op,ed;
18 pair<int,int> gird;
19 bool flag;
20
21 double Dis(Point p0,Point p1)
22 {
23     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
24                 (p0.second-p1.second)*(p0.second-p1.second));
25 }
26
27 double CalcDis(Point p0,Point p1,Point p2)
28 {
29     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
30 }
31
32 void build(int n,double w)
33 {
34     g.clear();
35     for (int i = 0;i < n;i++)
36         g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].
              second/w))].push_back(p[i]);
37 }
38
39 int main()
40 {
41     int t;
42     scanf("%d",&t);
43     for (int ft = 1;ft <= t;ft++)
44     {
45         scanf("%d",&n);
46         for (int i = 0;i < n;i++)
47         {
48             scanf("%lf%lf",&tx,&ty);
49             p[i] = make_pair(tx,ty);
50         }
51         random_shuffle(p,p+n);
52         ans = CalcDis(p[0],p[1],p[2]);
53         build(3,ans/2.0);
54         for (int i = 3;i < n;i++)
55         {
56             x = (int)floor(2.0*p[i].first/ans);
```

```
57              y = (int)floor(2.0*p[i].second/ans);
58              tmp.clear();
59              for (int k = 0;k < 9;k++)
60              {
61                  nx = x+step[k][0];
62                  ny = y+step[k][1];
63                  gird = make_pair(nx,ny);
64                  if (g.find(gird) != g.end())
65                  {
66                      op = g[gird].begin();
67                      ed = g[gird].end();
68                      for (it = op;it != ed;it++)
69                          tmp.push_back(*it);
70                  }
71              }
72              flag = false;
73              for (int j = 0;j < tmp.size();j++)
74                  for (int k = j+1;k < tmp.size();k++)
75                  {
76                      nowans = CalcDis(p[i],tmp[j],tmp[k]);
77                      if (nowans < ans)
78                      {
79                          ans = nowans;
80                          flag = true;
81                      }
82                  }
83              if (flag == true)
84                  build(i+1,ans/2.0);
85              else
86                  g[make_pair((int)floor(2.0*p[i].first/ans),(int)
                      floor(2.0*p[i].second/ans))].push_back(p[i]);
87          }
88          printf("%.3f\n",ans);
89      }
90  }
```

7.13 多圆面积并

7.13.1 去重

有时候可能需要去掉不需要的圆

```
1  for (int i = 0; i < n; i++)
2  {
3      scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
4      del[i] = false;
5  }
6  for (int i = 0; i < n; i++)
7      if (del[i] == false)
8      {
```

```
9           if (c[i].r == 0.0)  del[i] = true;
10          for (int j = 0; j < n; j++)
11              if (i != j)
12                  if (del[j] == false)
13                      if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j
                            ].r) <= 0)
14                          del[i] = true;
15      }
16  tn = n;
17  n = 0;
18  for (int i = 0; i < tn; i++)
19      if (del[i] == false)
20          c[n++] = c[i];
```

7.13.2 圆并

$ans[i]$表示被覆盖i次的面积

```
1   const double pi = acos(-1.0);
2   const double eps = 1e-8;
3   struct Point
4   {
5       double x,y;
6       Point(){}
7       Point(double _x,double _y)
8           {
9               x = _x;
10              y = _y;
11          }
12      double Length()
13          {
14              return sqrt(x*x+y*y);
15          }
16  };
17  struct Circle
18  {
19      Point c;
20      double r;
21  };
22  struct Event
23  {
24      double tim;
25      int typ;
26      Event(){}
27      Event(double _tim,int _typ)
28          {
29              tim = _tim;
30              typ = _typ;
31          }
```

```
32 };
33
34 int cmp(const double& a,const double& b)
35 {
36     if (fabs(a-b) < eps)    return 0;
37     if (a < b)  return -1;
38     return 1;
39 }
40
41 bool Eventcmp(const Event& a,const Event& b)
42 {
43     return cmp(a.tim,b.tim) < 0;
44 }
45
46 double Area(double theta,double r)
47 {
48     return 0.5*r*r*(theta-sin(theta));
49 }
50
51 double xmult(Point a,Point b)
52 {
53     return a.x*b.y-a.y*b.x;
54 }
55
56 int n,cur,tote;
57 Circle c[1000];
58 double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
59 Event e[4000];
60 Point lab;
61
62 int main()
63 {
64     while (scanf("%d",&n) != EOF)
65     {
66         for (int i = 0;i < n;i++)
67             scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
68         for (int i = 1;i <= n;i++)
69             ans[i] = 0.0;
70         for (int i = 0;i < n;i++)
71         {
72             tote = 0;
73             e[tote++] = Event(-pi,1);
74             e[tote++] = Event(pi,-1);
75             for (int j = 0;j < n;j++)
76                 if (j != i)
77                 {
78                     lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c
                        .y);
```

```
79                      AB = lab.Length();
80                      AC = c[i].r;
81                      BC = c[j].r;
82                      if (cmp(AB+AC,BC) <= 0)
83                      {
84                          e[tote++] = Event(-pi,1);
85                          e[tote++] = Event(pi,-1);
86                          continue;
87                      }
88                      if (cmp(AB+BC,AC) <= 0) continue;
89                      if (cmp(AB,AC+BC) > 0)  continue;
90                      theta = atan2(lab.y,lab.x);
91                      fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
92                      a0 = theta-fai;
93                      if (cmp(a0,-pi) < 0)    a0 += 2*pi;
94                      a1 = theta+fai;
95                      if (cmp(a1,pi) > 0)     a1 -= 2*pi;
96                      if (cmp(a0,a1) > 0)
97                      {
98                          e[tote++] = Event(a0,1);
99                          e[tote++] = Event(pi,-1);
100                         e[tote++] = Event(-pi,1);
101                         e[tote++] = Event(a1,-1);
102                     }
103                     else
104                     {
105                         e[tote++] = Event(a0,1);
106                         e[tote++] = Event(a1,-1);
107                     }
108                 }
109             sort(e,e+tote,Eventcmp);
110             cur = 0;
111             for (int j = 0;j < tote;j++)
112             {
113                 if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
114                 {
115                     ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
116                     ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(
                           pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
117                                       Point(c[i].c.x+c[i].r*cos
                                          (e[j].tim),c[i].c.y+c[
                                          i].r*sin(e[j].tim)))
                                          /2.0;
118                 }
119                 cur += e[j].typ;
120                 pre[cur] = e[j].tim;
121             }
122         }
```

```
123         for (int i = 1;i < n;i++)
124             ans[i] -= ans[i+1];
125         for (int i = 1;i <= n;i++)
126             printf("[%d] = %.3f\n",i,ans[i]);
127     }
128     return 0;
129 }
```

7.14 一个圆与多边形面积交

```
1  bool InCircle(Point a,double r)
2  {
3      return cmp(a.x*a.x+a.y*a.y,r*r) <= 0; //这里判断的时候EPS一定不要
           太小！！
4  }
5
6  double CalcArea(Point a,Point b,double r)
7  {
8      Point p[4];
9      int tot = 0;
10     p[tot++] = a;
11
12     Point tv = Point(a,b);
13     Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
14     Point near = LineToLine(Line(a,b),tmp);
15     if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
16     {
17         double A,B,C;
18         A = near.x*near.x+near.y*near.y;
19         C = r;
20         B = C*C-A;
21         double tvl = tv.x*tv.x+tv.y*tv.y;
22         double tmp = sqrt(B/tvl); //这样做只用一次开根
23         p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
24         if (OnSeg(Line(a,b),p[tot]) == true)    tot++;
25         p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
26         if (OnSeg(Line(a,b),p[tot]) == true)    tot++;
27     }
28     if (tot == 3)
29     {
30         if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length
               ()) > 0)
31             swap(p[1],p[2]);
32     }
33     p[tot++] = b;
34
35     double res = 0.0,theta,a0,a1,sgn;
36     for (int i = 0;i < tot-1;i++)
```

```
37      {
38          if (InCircle(p[i],r) == true && InCircle(p[i+1],r) ==
               true)
39          {
40              res += 0.5*xmult(p[i],p[i+1]);
41          }
42          else
43          {
44              a0 = atan2(p[i+1].y,p[i+1].x);
45              a1 = atan2(p[i].y,p[i].x);
46              if (a0 < a1)    a0 += 2*pi;
47              theta = a0-a1;
48              if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
49              sgn = xmult(p[i],p[i+1])/2.0;
50              if (cmp(sgn,0) < 0) theta = -theta;
51              res += 0.5*r*r*theta;
52          }
53      }
54      return res;
55  }
```

调用

```
1  area2 = 0.0;
2  for (int i = 0;i < resn;i++) //遍历每条边，按照逆时针
3      area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

7.15 精度问题

7.15.1 浮点数为啥会有精度问题

浮点数(以C/C++为准)，一般用的较多的是float、double。

	占字节数	数值范围	十进制精度位数
float	4	$-3.4e-38 \sim 3.4e38$	$6 \sim 7$
double	8	$-1.7e-308 \sim 1.7e308$	$14 \sim 15$

如果内存不是很紧张或者精度要求不是很低，一般选用double。14位的精度(是有效数字位，不是小数点后的位数)通常够用了。注意，问题来了，数据精度位数达到了14位，但有些浮点运算的结果精度并达不到这么高，可能准确的结果只有$10 \sim 12$位左右。那低几位呢？自然就是不可预料的数字了。这给我们带来这样的问题：即使是理论上相同的值，由于是经过不同的运算过程得到的，他们在低几位有可能(一般来说都是)是不同的。这种现象看似没太大的影响，却会一种运算产生致命的影响：==。恩，就是判断相等。注意，C/C++中浮点数的==需要完全一样才能返回true。

7.15.2 eps

eps缩写自epsilon，表示一个小量，但这个小量又要确保远大于浮点运算结果的不确定量。eps最常见的取值是$1e-8$左右。引入eps后，我们判断两浮点数a、b相等的方式如下:

```
int sgn(double a){return a < -eps ? -1 : a < eps ? 0 : 1;}
```

这样，我们才能把相差非常近的浮点数判为相等；同时把确实相差较大(差值大于eps)的数判为不相等。
养成好习惯，尽量不要再对浮点数做==判断。

7.15.3 eps带来的函数越界

如果sqrt(a), asin(a), acos(a) 中的a是你自己算出来并传进来的，那就得小心了。
如果a本来应该是0的，由于浮点误差，可能实际是一个绝对值很小的负数（比如$-1e-12$），这样sqrt(a)应得0的，直接因a不在定义域而出错。
类似地，如果a本来应该是± 1,则asin(a)、acos(a)也有可能出错。
因此，对于此种函数，必需事先对a进行校正。

7.15.4 输出陷阱I

现在考虑一种情况，题目要求输出保留两位小数。有个case的正确答案的精确值是0.005,按理应该输出0.01，但你的结果可能是0.005000000001(恭喜)，也有可能是0.004999999999(悲剧)，如果按照printf("%.2lf", a)输出，那你的遭遇将和括号里的字相同。
解决办法是，如果a为正，则输出$a + eps$, 否则输出$a - eps$

7.15.5 输出陷阱II

ICPC题目输出有个不成文的规定(有时也成文)，不要输出：-0.000
那我们首先要弄清，什么时候按printf("%.3lf", a)输出会出现这个结果。
直接给出结果好了：$a \in (-0.000499999\cdots, -0.000\cdots 1)$
所以，如果你发现a落在这个范围内，请直接输出0.000。更保险的做法是用sprintf直接判断输出结果是不是-0.000再予处理。

7.15.6 范围越界

请注意，虽然double可以表示的数的范围很大，却不是不穷大，上面说过最大是$1e308$。所以有些时候你得小心了，比如做连乘的时候，必要的时候要换成对数的和。

7.15.7 关于set

经观察,set不是通过==来判断相等的，是通过<来进行的，具体说来，只要$a < b$ 和$b < a$都不成立，就认为a和b相等，可以发现，如果将小于定义成:

```
bool operator < (const Dat dat)const{return val < dat.val - eps;}
```

就可以解决问题了。(基本类型不能重载运算符，所以封装了下)

7.15.8 输入值波动过大

这种情况不常见，不过可以帮助你更熟悉eps。假如一道题输入说，给一个浮点数a, $1e-20 < a < 1e20$。那你还敢用$1e-8$做eps么？合理的做法是把eps按照输入规模缩放到合适大小。

7.15.9 一些建议

容易产生较大浮点误差的函数有asin、 acos。欢迎尽量使用atan2。
另外，如果数据明确说明是整数，而且范围不大的话，使用int或者long long代替double都是极佳选择，因为就不存在浮点误差了

8 搜索

8.1 Dancing Links

8.1.1 估价函数

```
int h()
{
    bool vis[100];
    memset(vis,false,sizeof(vis));
    int i,j,k,res=0,mi,col;
    while(1)
    {
        mi=inf;
        for(i=R[head]; i!=head&&i<=2*n; i=R[i])
            if(mi>nk[i]&&!vis[i])
            {
                mi=nk[i];
                col=i;
            }
        if(mi==inf)
            break;
        res++;
        vis[col]=true;
        for(j=D[col]; j!=col; j=D[j])
            for(k=R[j]; k!=j; k=R[k])
            {
                if(C[k]>2*n)
                    continue;
                vis[C[k]]=true;
            }
    }
    return res;
}
```

8.1.2 DLX

```
void remove1(int col)
{
    int i,j;
    L[R[col]]=L[col];
    R[L[col]]=R[col];
    for(i=D[col];i!=col;i=D[i])
    {
        L[R[i]]=L[i];
        R[L[i]]=R[i];
    }
}
void remove2(int col)
```

```
13 {
14     int i,j;
15     L[R[col]]=L[col];
16     R[L[col]]=R[col];
17     for(i=D[col];i!=col;i=D[i])
18     {
19         for(j=R[i];j!=i;j=R[j])
20         {
21             U[D[j]]=U[j];
22             D[U[j]]=D[j];
23             --nk[C[j]];
24         }
25     }
26 }
27 void resume1(int col)
28 {
29     int i,j;
30     for(i=U[col];i!=col;i=U[i])
31     {
32         L[R[i]]=i;
33         R[L[i]]=i;
34     }
35     L[R[col]]=col;
36     R[L[col]]=col;
37 }
38 void resume2(int col)
39 {
40     int i,j;
41     for(i=U[col];i!=col;i=U[i])
42     {
43         for(j=L[i];j!=i;j=L[j])
44         {
45             ++nk[C[j]];
46             U[D[j]]=j;
47             D[U[j]]=j;
48         }
49     }
50     L[R[col]]=col;
51     R[L[col]]=col;
52 }
53 int h()
54 {
55   bool vis[100];
56   memset(vis,false,sizeof(vis));
57   int i,j,k,res=0,mi,col;
58   while(1)
59   {
60     mi=inf;
```

```
61      for(i=R[head];i!=head&&i<=2*n;i=R[i])
62        if(mi>nk[i]&&!vis[i])
63        {
64          mi=nk[i];
65          col=i;
66        }
67      if(mi==inf)
68        break;
69      res++;vis[col]=true;
70      for(j=D[col];j!=col;j=D[j])
71        for(k=R[j];k!=j;k=R[k])
72        {
73          if(C[k]>2*n)
74            continue;
75          vis[C[k]]=true;
76        }
77    }
78    return res;
79  }
80  bool DLX(int d,int deep)
81  {
82    if(d+h()>deep) return false;
83      if(R[head]==head||R[head]>2*n)
84        return true;
85      if(d>=deep)
86        return false;
87      int col,ma=inf;
88      int i,j;
89      for(i=R[head];i!=head&&i<=2*n;i=R[i])
90          if(nk[i]<ma)
91          {
92              col=i;
93              ma=nk[i];
94          }
95      remove1(col);
96      for(i=D[col];i!=col;i=D[i])
97      {
98          int flag=1;
99          for(j=R[i];;j=R[j])
100         {
101             if(j==R[i]&&!flag)
102                 break;
103             U[D[j]]=U[j];
104             D[U[j]]=D[j];
105             if(C[j]>2*n)
106                 remove2(C[j]);
107             else
108                 remove1(C[j]);
```

```
109             flag=0;
110         }
111         if(DLX(d+1,deep))
112           return true;
113         flag=1;
114         for(j=L[i];;j=L[j])
115         {
116             if(j==L[i]&&!flag)
117                 break;
118             if(C[j]>2*n)
119                 resume2(C[j]);
120             else
121                 resume1(C[j]);
122             U[D[j]]=j;
123             D[U[j]]=j;
124             flag=0;
125         }
126     }
127     resume1(col);
128     return false;
129 }
```

9 动态规划

9.1 斜率优化

```
#include<cstdio>
#include<algorithm>
using namespace std;
int a[1000],sum[1001],dp[1000][1000];
int deque[1000];
const int inf=0x7fffffff;
int N,s,t;
int calc(int i,int l,int j)//决策值计算
{
    return dp[j][l-1]-(sum[i]-sum[j])*(sum[N]-sum[i]);
}
bool check(int i,int l)//尾端判断
{
    int k1=deque[t-1],k2=deque[t-2];
    return (long long)(dp[k1][l]-dp[k2][l])*(sum[i]-sum[k1])>(
       long long)(dp[i][l]-dp[k1][l])*(sum[k1]-sum[k2]);
}
int main()
{
    int n,m;
    while (scanf("%d%d",&n,&m),n)
    {
        for (int i=0; i<n; i++)
            scanf("%d",&a[i]);
        N=n;
        sum[0]=0;
        for (int i=0; i<n; i++)
            sum[i+1]=sum[i]+a[i];
        dp[0][0]=0;
        for (int i=0; i<n; i++)
            for (int j=i+1; j<n; j++)
                dp[0][0]+=a[i]*a[j];
        for (int i=1; i<n; i++)
            dp[i][0]=inf;
        for (int i=1; i<n; i++)
        {
            dp[i][1]=inf;
            for (int j=0; j<i; j++)
                dp[i][1]=min(dp[i][1],calc(i,1,j));
        }
        for (int l=2; l<=m; l++)
        {
            s=t=0;//双端队列清空
            for (int i=l; i<n; i++)
```

```
44              {
45                  while (t-s>1 && check(i-1,l-1)) t--;
46                  deque[t++]=i-1;//决策加入
47                  while (t-s>1 && calc(i,l,deque[s])>calc(i,l,deque
                        [s+1])) s++;
48                  dp[i][l]=calc(i,l,deque[s]);
49              }
50          }
51          int ans=0x7fffffff;
52          for (int i=m; i<n; i++)
53              ans=min(ans,dp[i][m]);
54          printf("%d\n",ans);
55      }
56      return 0;
57  }
```

9.2 RMQ二版

```
1   void init()
2   {
3       int i,j;
4       int n=N,k=1,l=0;
5       for (i=0; i<n; i++)
6       {
7           f[i][0]=ele[i].num;
8           if (i+1>k*2)
9           {
10              k*=2;
11              l++;
12          }
13          lent[i+1]=l;
14      }
15      for (j=1; (1<<j)-1<n; j++)
16          for (i=0; i+(1<<j)-1<n; i++)
17              f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
18  }
19  int fint(int x,int y)
20  {
21      int k=lent[y-x+1];
22      return max(f[x][k],f[y-(1<<k)+1][k]);
23  }
```

9.3 二维LIS

```
1   #include<cstdio>
2   #include<map>
3   using namespace std;
4   map<int,int> mp[100001];
5   bool check(int idx,int x,int y)
```

```
{
    if (!idx) return 1;
    if (mp[idx].begin()->first>=x) return 0;
    map<int,int> ::iterator it=mp[idx].lower_bound(x);
    it--;
    if (it->second<y) return 1;
    else return 0;
}
int main()
{
    int n;
    scanf("%d",&n);
    int l=0,r=0;
    for (int i=0;i<n;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        int tl=l,tr=r;
        while (tl<tr)
        {
            int mid=(tl+tr+1)/2;
            if (check(mid,x,y))
                tl=mid;
            else
                tr=mid-1;
        }
        if (tl==r) r++;
        int idx=tl+1;
        map<int,int> ::iterator itl=mp[idx].lower_bound(x),itr=
           itl;
        while (itr!=mp[idx].end() && itr->second>y) itr++;
        if (mp[idx].find(x)!=mp[idx].end())
            y=min(y,mp[idx][x]);
        if (itl!=itr) mp[idx].erase(itl,itr);
        if (mp[idx].find(x)==mp[idx].end() || mp[idx][x]>y)
            mp[idx][x]=y;
    }
    printf("%d\n",r);
    return 0;
}
```

9.4 插头DP

Tower Defence独立插头+构造解
构造解的时候保存的是在hash_map的ele数组的下标位置
没想清楚千万别去写

```
int bit[12];

```

```
3  inline int getbit(long long sta,int pos)
4  {
5      return sta/bit[pos]%bit[1];
6  }
7
8  inline long long setbit(long long sta,int pos,int val)
9  {
10     return sta/bit[pos+1]*bit[pos+1]+val*bit[pos]+sta%bit[pos];
11 }
12
13 int n,m,mp[30][10];
14 char buf[30][10];
15 hash_map dp[2];
16 bool flag;
17 int key,val,upd,l,u,res,msk,cov,now,pr,resnow,resmsk,pru;
18 int w[15],s[15],top;
19 int pre[210][10007],preuse[210][10007];
20
21 void decode(int msk,int& key,int& cov)
22 {
23     int tmp;
24     key = cov = 0;
25     for (int i = 0; i < m+1; i++)
26     {
27         tmp = getbit(msk,i);
28         if (tmp > 0)
29         {
30             key = setbit(key,i,tmp-1);
31             cov = setbit(cov,i,1);
32         }
33     }
34 }
35
36 int encode(int key,int cov)
37 {
38     int res = 0,tmp;
39     for (int i = 0; i < m+1; i++)
40     {
41         tmp = getbit(cov,i);
42         if (tmp > 0)
43         {
44             tmp = getbit(key,i);
45             res = setbit(res,i,tmp+1);
46         }
47     }
48     return res;
49 }
50
```

```
51 void update(int a,int key,int cov,int val)
52 {
53     int msk = encode(key,cov);
54     int pos;
55     if (dp[a][msk] < val)
56     {
57         dp[a][msk] = val;
58         pos = dp[a].fint(msk);
59         pre[now][pos] = pr;
60         preuse[now][pos] = pru;
61     }
62 }
63
64 int count3(int sta)
65 {
66     int res = 0;
67     for (int i = 0; i < m+1; i++)
68         if (getbit(sta,i) == 3)
69             res++;
70     return res;
71 }
72
73 void expand(int sta)
74 {
75     top = 0;
76     for (int i = 0; i < m+1; i++)
77         if (getbit(sta,i) == 1)
78             s[top++] = i;
79         else if (getbit(sta,i) == 2)
80         {
81             w[s[top-1]] = i;
82             w[i] = s[top-1];
83             top--;
84         }
85 }
86
87 int main()
88 {
89     //freopen("TD.in","r",stdin);
90     //freopen("TDM.out","w",stdout);
91     bit[0] = 1;
92     for (int i = 1; i < 12; i++)  bit[i] = bit[i-1]*5;
93     int t;
94     scanf("%d",&t);
95     dp[0].init();
96     dp[1].init();
97     for (int ft = 1; ft <= t; ft++)
98     {
```

```
99          scanf("%d%d",&n,&m);
100         res = 0;
101         memset(mp,0,sizeof(mp));
102         memset(pre,0,sizeof(pre));
103         memset(preuse,0,sizeof(preuse));
104         for (int i = 0; i < n; i++)
105         {
106             scanf("%s",buf[i]);
107             for (int j = 0; j < m; j++)
108                 if (buf[i][j] == '.')
109                     mp[i][j] = 1;
110                 else if (buf[i][j] != 'B')
111                     mp[i][j] = 2;
112         }
113         dp[0].clear();
114         dp[1].clear();
115         flag = 0;
116         dp[flag][0] = 0;
117         int res = 0;
118         now = 0;
119         for (int i = 0; i < n; i++)
120         {
121             for (int j = 0; j < m; j++)
122             {
123                 dp[!flag].clear();
124                 for (int k = 0; k < dp[flag].N; k++)
125                 {
126                     msk = dp[flag].ele[k].key;
127                     pr = k;
128                     val = dp[flag].ele[k].val;
129                     decode(msk,key,cov);
130                     l = getbit(key,j);
131                     u = getbit(key,j+1);
132                     if (mp[i][j] == 0)//是障碍
133                     {
134                         if (l == 0 && u == 0)
135                         {
136                             pru = 0;
137                             update(!flag,key,setbit(setbit(cov,j
                                    ,0),j+1,0),val);
138                         }
139                     }
140                     else
141                     {
142                         if (mp[i][j] == 1 && l == 0 && u == 0)//
                                不要插
                                头
143                         {
```

```
144                             pru = 1;
145                             update(!flag,key,setbit(setbit(cov,j
                                    ,0),j+1,0),val);
146                         }
147                         if (getbit(cov,j) == 1 && l == 0)
                                continue;//不可以在这里搞插
                                头
148                         if (getbit(cov,j+1) == 1 && u == 0)
                                continue;
149                         cov = setbit(setbit(cov,j,1),j+1,1);//更新
                                覆盖情况
150                         upd = setbit(setbit(key,j,0),j+1,0);
151                         pru = 2;
152                         if (mp[i][j] == 2)
153                         {
154                             if (l == 0 && u == 0)
155                             {
156                                 if (count3(key) < 2)//可以新建独立
                                        插头
157                                 {
158                                     if (mp[i][j+1] != 0)
159                                         update(!flag,setbit(
                                                setbit(key,j,0),j+1,3)
                                                ,cov,val+1);
160                                     if (mp[i+1][j] != 0)
161                                         update(!flag,setbit(
                                                setbit(key,j,3),j+1,0)
                                                ,cov,val+1);
162                                 }
163                             }
164                             else if (l == 0 || u == 0)
165                             {
166                                 if (l+u < 3 && count3(key) < 2)//
                                        可以用一个独立插头来结束这条路
                                        径
167                                 {
168                                     expand(key);
169                                     if (l > 0)
170                                         update(!flag,setbit(upd,w
                                                [j],3),cov,val+1);
171                                     else
172                                         update(!flag,setbit(upd,w
                                                [j+1],3),cov,val+1);
173                                 }
174                                 else if (l+u == 3 && upd == 0)//
                                        路径的一
                                        端
175                                 {
```

```
                                if (res < val+1)
                                {
                                    res = val+1;
                                    resnow = now-1;
                                    resmsk = k;
                                }
                            }
                        }
                        else if (l == 0 && u == 0)
                        {
                            if (mp[i][j+1] != 0 && mp[i+1][j] !=
                               0)//可以新建插
                               头
                                update(!flag,setbit(setbit(key,j
                                   ,1),j+1,2),cov,val+1);
                        }
                        else if (l == 0 || u == 0)
                        {
                            if (mp[i][j+1] != 0)//可以延续插头
                                update(!flag,setbit(upd,j+1,l+u),
                                   cov,val+1);
                            if (mp[i+1][j] != 0)//可以延续插头
                                update(!flag,setbit(upd,j,l+u),
                                   cov,val+1);
                        }
                        else if (l == u)
                        {
                            if (l < 3) //合并两个相同的括号
                            {
                                expand(key);
                                if (l == 1)
                                    update(!flag,setbit(upd,w[j
                                       +1],1),cov,val+1);
                                else
                                    update(!flag,setbit(upd,w[j
                                       ],2),cov,val+1);
                            }
                            else if (upd == 0)//合并两个独立插头
                            {
                                if (res < val+1)
                                {
                                    res = val+1;
                                    resnow = now-1;
                                    resmsk = k;
                                }
                            }
                        }
```

```
                        else if (l == 3 || u == 3)//合并独立插头与括
                            号
                        {
                            expand(key);
                            if (l == 3)
                                update(!flag,setbit(upd,w[j+1],3)
                                    ,cov,val+1);
                            else
                                update(!flag,setbit(upd,w[j],3),
                                    cov,val+1);
                        }
                        else if (l == 2 || u == 1) //合并)(
                            update(!flag,upd,cov,val+1);
                    }
                }
                flag = !flag;
                now++;
            }
            if (i+1 == n)   break;

            dp[!flag].clear();
            for (int k = 0; k < dp[flag].N; k++)
            {
                msk = dp[flag].ele[k].key;
                pr = k;
                val = dp[flag].ele[k].val;
                pru = 0;
                decode(msk,key,cov);
                update(!flag,key*bit[1],cov*bit[1],val);
            }
            now++;
            flag = !flag;
        }

        printf("Case␣%d:␣%d\n",ft,res);
        for (int i = resnow; i >= 0; i--)
        {
            if (preuse[i][resmsk] == 1)
                buf[i/(m+1)][i%(m+1)] = 'W';
            resmsk = pre[i][resmsk];
        }
        for (int i = 0; i < n; i++)
            printf("%s\n",buf[i]);
        printf("\n");
    }
    return 0;
}
```

10 杂物

10.1 高精度数

支持乘以整数和加法。

```
struct BigInt
{
    const static int mod = 100000000;
    int a[600],len;
    BigInt (){}
    BigInt (int v)
    {
        len = 0;
        do
        {
            a[len++] = v%mod;
            v /= mod;
        }while(v);
    }
    BigInt operator *(const int& b) const
    {
        BigInt res;
        res.len = len;
        for (int i = 0; i <= len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < len; ++i)
        {
            res.a[i] += a[i]*b;
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
        if (res.a[len] > 0)  res.len++;
        return res;
    }
    BigInt operator +(const BigInt& b) const
    {
        BigInt res;
        res.len = max(len,b.len);
        for (int i = 0; i <= res.len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < res.len; ++i)
        {
            res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0)
                ;
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
```

```
42          if (res.a[res.len] > 0)  res.len++;
43          return res;
44      }
45      void output()
46      {
47          printf("%d",a[len-1]);
48          for (int i = len-2; i >= 0; --i)
49              printf("%08d",a[i]);
50          printf("\n");
51      }
52  };
```

10.2 整数外挂

```
1   int wg;
2   char ch;
3   bool ng;
4
5   inline int readint()
6   {
7       ch = getchar();
8       while (ch != '-' && (ch < '0' || ch > '9')) ch = getchar();
9       if (ch == '-')
10      {
11          ng = true;
12          ch = getchar();
13      }
14      else
15          ng = false;
16      wg = ch-'0';
17      ch = getchar();
18      while (ch >= '0' && ch <= '9')
19      {
20          wg = wg*10+ch-'0';
21          ch = getchar();
22      }
23      if (ng == true) wg = -wg;
24      return wg;
25  }
```

10.3 Java

10.3.1 文件操作

```
1   import java.io.*;
2   import java.util.*;
3   import java.math.*;
4   import java.text.*;
5
```

```
6  public class Main
7  {
8
9      public static void main(String args[]) throws
           FileNotFoundException, IOException
10     {
11         Scanner sc = new Scanner(new FileReader("a.in"));
12         PrintWriter pw = new PrintWriter(new FileWriter("a.out"))
               ;
13         int n,m;
14         n=sc.nextInt();//读入下一个INT
15         m=sc.nextInt();
16
17         for(ci=1; ci<=c; ++ci)
18         {
19             pw.println("Case␣#"+ci+":␣easy␣for␣output");
20         }
21
22         pw.close();//关闭流并释放，这个很重要，否则是没有输出的
23         sc.close();//关闭流并释放
24     }
25 }
```

10.3.2 优先队列

```
1  PriorityQueue queue = new PriorityQueue( 1, new Comparator()
2  {
3      public int compare( Point a, Point b )
4      {
5    if( a.x < b.x || a.x == b.x && a.y < b.y )
6        return -1;
7    else if( a.x == b.x && a.y == b.y )
8        return 0;
9    else
10       return 1;
11     }
12 });
```

10.3.3 Map

```
1  Map map = new HashMap();
2  map.put("sa","dd");
3  String str = map.get("sa").toString;
4
5  for(Object obj : map.keySet()){
6      Object value = map.get(obj );
7  }
```

10.3.4 sort

```
static class cmp implements Comparator
{
    public int compare(Object o1,Object o2)
    {
  BigInteger b1=(BigInteger)o1;
  BigInteger b2=(BigInteger)o2;
  return b1.compareTo(b2);
    }
}
public static void main(String[] args) throws IOException
{
    Scanner cin = new Scanner(System.in);
    int n;
    n=cin.nextInt();
    BigInteger[] seg = new BigInteger[n];
    for (int i=0;i<n;i++)
  seg[i]=cin.nextBigInteger();
    Arrays.sort(seg,new cmp());
}
```

10.4 hashmap

```
struct hash_map
{
    const static int mod=10007;
    int head[mod];
    struct hash_tables
    {
        int key;
        int val;
        int next;
    } ele[10007];
    int N;
    int getHash(int x)
    {
        return x%mod;
    }
    void init()
    {
        memset(head,255,sizeof(head));
        N=0;
    }
    void clear()
    {
        for (int i = 0; i < N; i++)
            head[getHash(ele[i].key)] = -1;
        N = 0;
    }
    int fint(int x)
```

```
28      {
29          for (int i=head[getHash(x)]; i!=-1; i=ele[i].next)
30              if (ele[i].key==x) return i;
31          return -1;
32      }
33      void insert(int x)
34      {
35          int tmp=getHash(x);
36          ele[N].key=x;
37          ele[N].val=0;
38          ele[N].next=head[tmp];
39          head[tmp]=N++;
40      }
41      int& operator [](int x)
42      {
43          int tmp=fint(x);
44          if (tmp==-1)
45          {
46              insert(x);
47              return ele[N-1].val;
48          }
49          else
50              return ele[tmp].val;
51      }
52  };
```

10.5 C++&STL常用函数

10.5.1 lower_bound/upper_bound

不解释

```
1  iterator lower_bound(const key_type &key )\\返回一个迭代器，指向键值>=
       key的第一个元素。
2  iterator upper_bound(const key_type &key )\\返回一个迭代器，指向键值>
       key的第一个元素。
3
4  #include <iostream>
5  #include <algorithm>
6  #include <vector>
7  using namespace std;
8
9  int main () {
10   int myints[] = {10,20,30,30,20,10,10,20};
11   vector<int> v(myints,myints+8);           // 10 20 30 30 20 10
         10 20
12   vector<int>::iterator low,up;
13
```

```
14    sort (v.begin(), v.end());                  // 10 10 10 20 20 20
         30 30
15
16    low=lower_bound (v.begin(), v.end(), 20); //          ^
17    up= upper_bound (v.begin(), v.end(), 20); //
         ^
18
19    cout << "lower_bound␣at␣position␣" << int(low- v.begin()) <<
         endl;
20    cout << "upper_bound␣at␣position␣" << int(up - v.begin()) <<
         endl;
21
22    return 0;
23  }
```

Output:

```
1  lower_bound at position 3
2  upper_bound at position 6
```

10.5.2 rotate

把数组后一半搬到前面

```
1  template <class ForwardIterator>
2    void rotate ( ForwardIterator first, ForwardIterator middle,
3                  ForwardIterator last );
```

10.5.3 nth_element

```
1  template <class RandomAccessIterator>
2    void nth_element ( RandomAccessIterator first,
        RandomAccessIterator nth,
3                       RandomAccessIterator last );
4
5  template <class RandomAccessIterator, class Comapre>
6    void nth_element ( RandomAccessIterator first,
        RandomAccessIterator nth,
7                       RandomAccessIterator last, Compare comp );
```

10.5.4 bitset

取用

```
1  bitset<4> mybits;
2
3  mybits[1]=1;              // 0010
4  mybits[2]=mybits[1];      // 0110
```

翻转

```
bitset<4> mybits (string("0001"));

cout << mybits.flip(2) << endl;      // 0101
cout << mybits.flip() << endl;       // 1010
```

运算

```
bitset<4> first (string("1001"));
bitset<4> second (string("0011"));

cout << (first^=second) << endl;          // 1010 (XOR,assign)
cout << (first&=second) << endl;          // 0010 (AND,assign)
cout << (first|=second) << endl;          // 0011 (OR,assign)

cout << (first<<=2) << endl;              // 1100 (SHL,assign)
cout << (first>>=1) << endl;              // 0110 (SHR,assign)

cout << (~second) << endl;                // 1100 (NOT)
cout << (second<<1) << endl;              // 0110 (SHL)
cout << (second>>1) << endl;              // 0001 (SHR)

cout << (first==second) << endl;          // false (0110==0011)
cout << (first!=second) << endl;          // true  (0110!=0011)

cout << (first&second) << endl;           // 0010
cout << (first|second) << endl;           // 0111
cout << (first^second) << endl;           // 0101
```

10.5.5 multimap

遍历

```
multimap<char,int> mymm;
multimap<char,int>::iterator it;
char c;

mymm.insert(pair<char,int>('x',50));
mymm.insert(pair<char,int>('y',100));
mymm.insert(pair<char,int>('y',150));
mymm.insert(pair<char,int>('y',200));
mymm.insert(pair<char,int>('z',250));
mymm.insert(pair<char,int>('z',300));

for (c='x'; c<='z'; c++)
{
  cout << "There are " << (int)mymm.count(c);
```

```
15    cout << "␣elements␣with␣key␣" << c << ":";
16    for (it=mymm.equal_range(c).first; it!=mymm.equal_range(c).
         second; ++it)
17      cout << "␣" << (*it).second;
18    cout << endl;
19  }
20  /*
21  Output:
22
23  There are 1 elements with key x: 50
24  There are 3 elements with key y: 100 150 200
25  There are 2 elements with key z: 250 300
26  */
```

二分查找

```
1   multimap<char,int> mymultimap;
2   multimap<char,int>::iterator it,itlow,itup;
3
4   mymultimap.insert(pair<char,int>('a',10));
5   mymultimap.insert(pair<char,int>('b',121));
6   mymultimap.insert(pair<char,int>('c',1001));
7   mymultimap.insert(pair<char,int>('c',2002));
8   mymultimap.insert(pair<char,int>('d',11011));
9   mymultimap.insert(pair<char,int>('e',44));
10
11  itlow=mymultimap.lower_bound ('b');  // itlow points to b
12  itup=mymultimap.upper_bound ('d');   // itup points to e (not d)
13
14  // print range [itlow,itup):
15  for ( it=itlow ; it != itup; it++ )
16    cout << (*it).first << "␣=>␣" << (*it).second << endl;
17
18  /*
19  Output:
20
21  b => 121
22  c => 1001
23  c => 2002
24  d => 11011
25  */
```

删除

```
1   multimap<char,int> mymultimap;
2   multimap<char,int>::iterator it;
3
4   // insert some values:
5   mymultimap.insert(pair<char,int>('a',10));
```

```
mymultimap.insert(pair<char,int>('b',20));
mymultimap.insert(pair<char,int>('b',30));
mymultimap.insert(pair<char,int>('c',40));
mymultimap.insert(pair<char,int>('d',50));
mymultimap.insert(pair<char,int>('d',60));
mymultimap.insert(pair<char,int>('e',70));
mymultimap.insert(pair<char,int>('f',80));

it=mymultimap.find('b');
mymultimap.erase (it);                        // erasing by iterator (1 element)

mymultimap.erase ('d');                       // erasing by key (2 elements)

it=mymultimap.find ('e');
mymultimap.erase ( it, mymultimap.end() ); // erasing by range

// show content:
for ( it=mymultimap.begin() ; it != mymultimap.end(); it++ )
  cout << (*it).first << " => " << (*it).second << endl;

/*
Output:

a => 10
b => 30
c => 40
*/
```

10.6 位运算

10.6.1 基本操作

注意括号

10.6.2 枚举长为n含k个1的01串

```
int n = 5,k = 3;
for (int s = (1 << k)-1,u = 1 << n; s < u;)
{
    for (int i = 0;i < n;i++)
        printf("%d",(((s>>(n-1-i))&1) == 1));
    printf("\n");

    int b = s & -s;
    s = (s+b)|(((s^(s+b))>>2)/b);
}
```

功能	示例	位运算
去掉最后一位	(101101 → 10110)	x shr 1
在最后加一个0	(101101 → 1011010)	x shl 1
在最后加一个1	(101101 → 1011011)	x shl 1+1
把最后一位变成1	(101100 → 101101)	x or 1
把最后一位变成0	(101101 → 101100)	x or 1-1
最后一位取反	(101101 → 101100)	x xor 1
把右数第k位变成1	(101001 → 101101, $k = 3$)	x or (1 shl (k-1))
把右数第k位变成0	(101101 → 101001, $k = 3$)	x and not (1 shl (k-1))
右数第k位取反	(101001 → 101101, $k = 3$)	x xor (1 shl (k-1))
取末三位	(1101101 → 101)	x and 7
取末k位	(1101101 → 1101, $k = 5$)	x and (1 shl k-1)
取右数第k位	(1101101 → 1, $k = 4$)	x shr (k-1) and 1
把末k位变成1	(101001 → 101111, $k = 4$)	x or (1 shl k-1)
末k位取反	(101001 → 100110, $k = 4$)	x xor (1 shl k-1)
把右边连续的1变成0	(100101111 → 100100000)	x and (x+1)
把右起第一个0变成1	(100101111 → 100111111)	x or (x+1)
把右边连续的0变成1	(11011000 → 11011111)	x or (x-1)
取右边连续的1	(100101111 → 1111)	(x xor (x+1)) shr 1
去掉右起第一个1的左边	(100101000 → 1000)	x and (x xor (x-1))

10.7 其它

10.7.1 对跑脚本

```
while true; do
  ./gen > input
  ./sol < input > output.sol
  ./bf < input > output.bf

  diff output.sol output.bf
  if [ $? -ne 0 ] ; then break; fi
done
```